ANO 1 Nº 04

Cz$ 660,00

CPU

Inteligência Artificial

Screen ϕ

Plotador Gráfico

Arquivos de Lote

ÁGUIA INFORMATICA LTDA.
AV. N. SRA. DE COPACABANA 605/804
COPACABANA
RIO DE JANEIRO - RJ
CEP 22040
TEL: (021)235-3541
TELEX: 2138953

DIRETOR RESPONSÁVEL
GONÇALO MURTEIRA

DIRETORIA TÉCNICA
ANTONIO F. S. SHALDERS
CARLOS E. A. MOREIRA
ANDRÉ L. DE FREITAS
J. L. FONSECA

JORNALISTA RESPONSÁVEL
DOLAR TANUS
REGISTRO 430-RS

REVISÃO DE TEXTO
LAURA MARIA PINTO

CAPA
JOSÉ AGUILERA

ASSINATURAS
EDUARDO SIMPLÍCIO

ADMINISTRAÇÃO
JOSÉ A. NASCIMENTO

PROJETO GRÁFICO
LUCIANA MONTENEGRO

IMPRESSÃO
EDITORA LUA NOVA

O mercado do MSX no Brasil vem, de certa forma, apresentando um crescimento numa época em que a situação econômica do país não permite tal façanha.

Na área de hardware, a linha MSX já conta com drives de 3 1/2" há algum tempo e, recentemente, ganhou um aliado na sua luta pela profissionalização, que vem a ser a interface RS232 lançada pela Cibertron.

Editoras de livros técnicos também vêm dando maior atenção para os usuários MSX e sempre estão lançado títulos novos na praça.

Neste número de CPU, temos a certeza de deixá-lo a par do que está acontecendo no mundo do MSX e fornecer-lhe o maior número de informações sobre este potente computador.

Abrindo este número, um assunto que é tão pouco explorado no Brasil, mas é por demais fascinante: A Inteligência Artificial.

GONÇALO MURTEIRA

ÍNDICE

SEÇÕES

MSX NEWS

DACAL INFORMÁTICA - SOFTWARES EDUCACIONAIS

Com o crescente desenvolvimento da tecnologia, o micro computador pessoal veio atender a necessidade de reformulação da metodologia de ensino/aprendizagem tradicional.

Naturalmente, o computador não substitui o professor ou os pais e não supre a relação humana necessária à formação de todo ser, mas ajudam os estudantes a avançarem no seu próprio ritmo, tornando a aquisição de determinadas habilidades.

Hoje, com programas específicos para a pré-escola e primeiro grau, os computadores estimulam a criança, julgam as suas respostas, ajudam a descobrir novos caminhos sem frustá-las, com uma imensa vantagem sobre o ser humano: sempre podem, pacientemente, retornar ao ponto de partida.

São considerados programas educacionais nesta fase apenas aqueles que ajudam a criança a desenvolver uma habilidade ou condição que lhe permite posterior aprendizado formal.

O programa pedagógico deve ter um objetivo claro e definido, ser atraente, estimulante e ,ao mesmo tempo, transmitir à criança toda a tranquilidade necessária para que possa usá-lo sem receio de ser castigada por algum comportamento incorreto, sem as agitações do vídeo game.

Esta é a linha de pensamento da Dacal Informática, uma empresa relativamente nova e que já tem para venda alguns softwares educacionais desenvolvidos, sendo um deles o programa MATIZ, que faz parte de um conjunto de oito desenvolvidos para crianças com idade a partir dos 3 anos e que se encontram na pré-escola, tendo o objetivo específico de desenvolver a percepção visual de formas e cores.

O programa possui telas para orientar o educador ou os pais a respeito dos objetivos e usos do programa.

Para cada acerto feito pela criança, uma carinha sorrindo surgirá na parte inferior do vídeo, que também é um contador. Após um certo número de acertos e algumas carinhas acumuladas, o programa automaticamente muda de nível.

Maiores detalhes sobre os softwares desenvolvidos pela Dacal Informática, que tem como responsável Vera da Cal, formada em Pedagogia pela UFP e Psicopedagogia pela PUC, poderão ser obtidas através dos telefones:

011-216.6944 em São Paulo, ou
021-712.4393 no Rio de Janeiro.

INTERFACE RS232/TERMINAL

A Cibertron está lançando no mercado uma Interface RS232 que permite a emulação de um terminal de IBM, ou seja, possibilita a conexão de micros do padrão MSX a um IBM PC. Assim, através de um software específico, um único PC poderá rodar ao mesmo tempo vários programas, como, por exemplo, um processador de texto, banco de dados, ou ainda permitir a sua utilização em colégios e cursos onde os alunos utilizariam o MSX como terminal e um IBM PC seria utilizado pelo professor.

A utilização de um MSX com terminal de um IBM PC também tem suas vantagens evidenciadas se o sistema for utilizado em Empresas, devido ao baixo custo que um computador da linha MSX apresenta em relação aos demais terminais existentes no mercado.

A interface da Cibertron também permite a conexão do MSX à rede Cirandão, da Embratel, e ao Vídeo Texto, da Telesp, além dos demais serviços deste gênero disponíveis, como, por exemplo, o Sampa.Também será possível ao usuário desta interface a comunicação, com certa facilidade, micro a micro.

Maiores detalhes deste lançamento poderão ser obtidos diretamente na Cibertron, através do telefone 011-298.3299 ou através do telex de número 1163112

ASTROLOGIA

A Editora Aleph mais uma vez supreende os usuários da linha MSX lançando um livro que, pelo menos para o pessoal da revista, já despertou o maior interesse.

O livro ASTROLOGIA já se encontra em fase final de edição, devendo estar brevemente nas lojas, segundo informações fornecidas pelo Prof. Pierluigi Piazzi, responsável pela editora.

A Editora Aleph, como já é do conhecimento de vocês, lançou os livros "Aprofundando-se no MSX" e "Cem dicas para MSX" que, a nosso ver, são instrumentos de consulta indispensáveis para os usuários, tanto para os iniciantes como para os de nível de programação mais avançado, além de outros, também de excelente nível de interesse.

Portanto, tudo nos leva a acreditar que, além do título "Astrologia", que por si só já é empolgante, este será mais um best seller da Aleph.

MSX WORD

Recebemos em nossa redação uma cópia e respectivo manual da nova versão do mais famoso e utilizado processador de textos para a linha MSX WORD enviado pelo seu fabricante, a Cibertron Software.

Vejas maiores detalhes na seção de análise de software.

INTELIGÊNCIA ARTIFICIAL EM PASCAL

As incríveis façanhas dos computadores Hal e Sal, do filme 2010 não estão longe de acontecerem na realidade . Neste artigo, os autores dão noções do uso da inteligência artificial no MSX, fornecendo um programa exemplo em Pascal.

ANTONIO F.S.SHALDERS
VICTOR ELIASZ WELMAN

Programas que simulam o raciocínio humano já são uma realidade, deixando os cenários de filmes de ficção científica, como por exemplo os computadores HAL e SAL, do filme "2010". Tais computadores eram capazes de incríveis façanhas, tal como reconhecimento e síntese de voz, e um incrível poder de dedução e raciocínio.

É lógico que o que propomos neste artigo não chega nem aos pés disso, mas não deixa de ser muito interessante, principalmente se for levado em conta que não foi usada nenhuma linguagem específica para programação em IA, como o LISP e o PROLOG, sendo realizado totalmente em Pascal.

Um programa em IA simula de algum modo o pensamento humano (ou pelo menos tenta), fazendo que o próprio programa tome uma decisão ou então faça uma dedução sobre algum fato.

Você que sempre escuta de seus pais e colegas: "O que é que você tanto faz na frente daquela máquina burra ?!" terá chance de se defender, mostrando-lhes que o seu computador não é uma máquina assim tão burra, e que, se corretamente programado, pode até "aprender". Caso o programa cometa algum erro, este aprende a não fazê-lo novamente.

Um caso típico de programas que utilizam técnicas de IA são os jogos de tabuleiro como o xadrez e o gamão. As primeiras partidas são facilmente vencidas por você, mas, depois de um certo tempo, fica praticamente impossível vencer a máquina, pois o jogo "aprende" a não perder, com seus próprios erros.

O que podemos fazer em Pascal é simularmos matematicamente alguns procedimentos que facilitam a programação em IA.

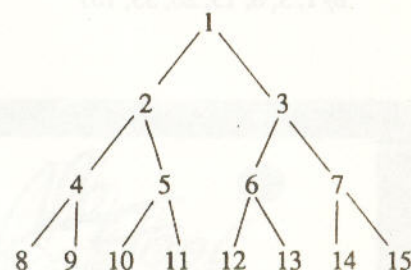
O algoritmo utilizado é o da árvore (sem podas), por ser o de mais fácil implementação, mas nem por isso o programa deixa de ser interessante, pois o mesmo superou todas as expectativas em relação ao desempenho.

O método da árvore consiste em seguirmos um determinado caminho (galho) até chegarmos a uma resposta coerente (fruto).

O programa mostrado é conhecido tecnicamente por reconhecedor. No nosso caso, como o próprio nome indica, é capaz de reconhecer uma pessoa baseado em suas características ou em fatos a ela referentes.

A situação correspondente na vida humana é a seguinte: um amigo seu lhe pergunta se você conhece uma determinada pessoa. A priori, você não se lembra e vai, então, fazendo uma série de perguntas que o ajudarão a reconhecê-la. O reconhecimento, neste caso, dá-se quando uma imagem mental da pessoa em questão é formada no seu consciente.

Caso você tenha pensado em outra pessoa diferente, irá procurar alguma coisa que diferencie a pessoa em questão da que você pensou, de modo a não cometer o mesmo erro novamente, certo ?

Analisemos, agora, como é constituída uma árvore. Para isso, observe a figura 1 atentamente.

O número 1 no topo da árvore é a semente (ou pergunta inicial). A partir desta pergunta, podemos distinguir até oito frutos (que estão na base da árvore). Os frutos são as terminações dos galhos.

Note que há apenas um caminho possível para chegarmos a cada fruto e é isso que nos permite identificá-lo corretamente.

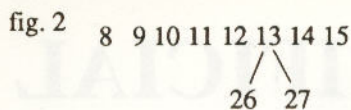
Suponha que nos "nós" da árvore (chamaremos de "nós" os pontos por onde passam dois ou mais galhos) hajam perguntas e cada vez que a resposta for "sim", você ande para a esquerda, e quando for "não", ande para a direita.

Por exemplo: para chegarmos ao fruto 13, devemos responder não à pergunta do nó 1, sim à do no 3 e não à do nó 6. Chegaremos, então, ao fruto em questão.

Suponha, agora, que a resposta obtida em 13 esteja errada. Devemos, então, converter este fruto em um nó, com uma pergunta que nos permita distinguir a resposta certa da obtida, e criar dois frutos, um com a resposta certa e outro com a obtida.

No caso mencionado, as últimas linhas da árvore ficarão como as mostradas na figura 2.

fig. 2

8 9 10 11 12 13 14 15
/ \
26 27

Para construirmos uma árvore, basta escolhermos uma semente inicial (1) e ir multiplicando este valor por dois para uma resposta positiva, e multiplicando por dois e somando um para uma resposta negativa.

Concluímos, então, que um caminho é formado por números que são o dobro, ou o dobro mais um, do número anterior ao mesmo, no caminho.

Este é o procedimento matemático adotado pelo algoritmo da árvore (sem podas). São exemplos de caminhos:

a) 1, 2, 4, 8, 17, 34, 69
b) 1, 3, 6, 13, 26, 53, 107

O que o nosso programa faz é estruturar uma árvore deste tipo para então seguir os galhos até chegar a um fruto coerente.

No programa, a semente é a pergunta "É homem ?" e os dois frutos iniciais são "João", para uma resposta afirmativa, e "Maria" caso a reposta seja negativa.

Você deve substituir estes dois nomes por outros de sua conveniência.

Se o programa for utilizado com o Turbo Pascal em opção de compilação em memória, o array responsável pelo número de nomes não deverá ser superior a 220. Na opção de compilação em disco, não deverá ser superior a 840.

Este programa foi elaborado para reconhecer pessoas, mas nada impede que seja usado em áreas profissionais, como por exemplo na área médica.

Neste caso, se você é médico, poderá colocar em cada nó um sintoma e nos frutos um pré-diagnóstico.

É possivel aplicarmos o programa em robótica.

Neste caso, suponha que você quer fazer com que um robô pegue um determinado objeto em sua casa e o leve até você. O robô teria um mapa de sua casa e iria lhe perguntando a respeito dos locais onde o objeto em questão estaria, selecionando, assim, o local exato para ir até lá e trazer-lhe o tal objeto.

As possibilidades de uso de IA são ilimitadas.

Há casos de programas para geoprospecção, estrategistas militares, diagnóstico de doenças, e muitos outros tipos.

```
program intan;

{otimização de arrays para
velocidade}
{$X-}

{área de definição de tipos}

type fi=string[40];
type pd=record
     k:integer;
     c:real;
     p:fi;
     end;

{área de definição de variáveis}

var
     p           : array[0..250] of
pd;  {array do número de pessoas}
     d,te,b      : pd;
     re,s        : string[3];
     ar,tr,aqv_arq : fi;
     co          : real;
     j,l,h,m,s   : integer;
     aqv         : file of pd;

{le um caractere do teclado}

function gr:char;
  var rp:char;
  begin
    readln(rp);
    gr:=upcase(rp);
  end;

function rx(x:integer):real;
  var i :integer;
      ax:real;
  begin
    ax:=1;
    for i:=1 to x do
      ax:=10*ax;
    rx:=ax;
  end;

{dobrar}

function db(x:real):real;
  begin
    a:=trunc(ln(x)/ln(10));
    if a<=9 then
      db:=2*x
    else
      db:=rx(a-9)*x
  end;

{dobrar mais um}

function dm(x:real):real;
  begin
    a:=trunc(ln(x)/ln(10));
    if a<=9 then
      dm:=2*x+1
    else
      dm:=rx(a-8)*x
  end;

{achar nomes}

function achar(x:real):integer;
  var ya:integer;
  begin
    ya:=1;
    while p[ya].c<x do
      ya:=ya+1;
    achar:=ya;
  end;

{ler dados do disco}

procedure lds;
  begin
    j:=0;
    assign(aqv,aqv_arq);
    reset(aqv);
    while not eof(aqv) do
      begin
        read(aqv,d);
        p[j]:=d;
        j:=j+1;
      end;
    j:=j-1;
    p[0]:=b;
  end;

{gravar no disco}

procedure gds;
  begin
    assign(aqv,aqv_arq);
    rewrite(aqv);
    for h:=0 to j do
      write(aqv,p[h]);
    close(aqv);
  end;

procedure ir(ss:fi;rr:real);
  var i,v :integer;
  begin
    i:=0;
    while ((p[i].c<rr)and(i<=j)) do
      i:=i+1;
    for v:=j downto i do
      p[v+1]:=p[v];
    p[i].c:=rr;
    p[i].k:=2;
    p[i].p:=ss;
    j:=j+1;
  end;

{controle principal de processos}

procedure ctr;
  begin
    l:=0;
    while l<=j do
      begin
        te:=p[l];
        if te.k=1 then
          begin
            writeln(te.p,' ?');
            re:=gr;
            if re=s then co:=db(te.c)
            else co:=dm(te.c);
            l:=achar(co);
          end;
        if te.k=2 then
          begin
            writeln('Por acaso é: ');
            writeln(te.p,' ?');
            re:=gr;
            if re<>s then
              begin
                writeln('Quem é ?');
                readln(ar);
                writeln('Dê uma
diferença entre ');
                writeln(te.p,' e ',ar);
                readln(tr);
                writeln(te.p);
                writeln(tr, ' ?');
                re:=gr;
                if re<>s then
                  begin
                    co:=db(te.c);
                    te.c:=dm(te.c)
                  end
                else
                  begin
                    co:=dm(te.c);
                    te.c:=db(te.c);
                  end;
                p[l].k:=1;
                p[l].p:=tr;ir(te.p,te.c);
                ir(ar,co);
              end;
            l:=j+1;
          end;
      end;
  end;

{inicialização do vetor}

procedure init;
  var v:string[11];
  begin
    writeln('Carregar arquivo ?');
    v:=gr;
    if v=s then
      begin
        writeln('Nome do arquivo
?');
        readln(v);
        clrscr;
        aqv_arq:=v;
        lds;
      end;
    writeln;
  end;

{finalização do programa}

procedure final;
  var v:string[11];
  begin
    writeln('Salvar arquivo ?');
    v:=gr;
    if v=s then
      begin
        writeln('Nome do arquivo
?');
        readln(v);
        aqv_arq:=v;
        gds;
      end;
  end;

{apresentação}

procedure apresentação;
  begin
    clrscr;
    writeln('------------
------------');
    writeln('   IntAn   1.00');
    writeln('   (C) 1988
by');
    writeln('   Victor E.
Welman');
    writeln('      e');
    writeln('   A. F.
Shalders');
    writeln('------------
------------');
  end;

{corpo do programa principal}

begin
  apresentação;
  gotoxy(1,10);
  p[0].k:=1;
  p[0].c:=1;
  p[0].p:='É HOMEM';
  b:=p[0];
  p[1].k:=2;
  p[1].c:=2;
  {*** semente da parte
  masculina ***}
  p[1].p:='Joao';

  p[2].k:=2;
  p[2].c:=3;
  {*** semente da parte
  feminina ***}
  p[2].p:='Maria';

  j:=2;
  s:='S';
  init;
  m:=1;
  while m>0 do
    begin
      writeln('Pense em alguém:');
      ctr;
      writeln('Mais alguém ? ');
      re:=gr;
      if re<>s then m:=0;
    end;
  final;
end.
```

O COMANDO DRAW

PROF. PERLUIGI PIAZI

O comando DRAW do BASIC MSX, na realidade, é uma macro linguagem gráfica que permite a elaboração de desenhos complexos nas telas 2 e 3.

Sua sintaxe é extremamente simples

DRAW (expressão ou variável string)

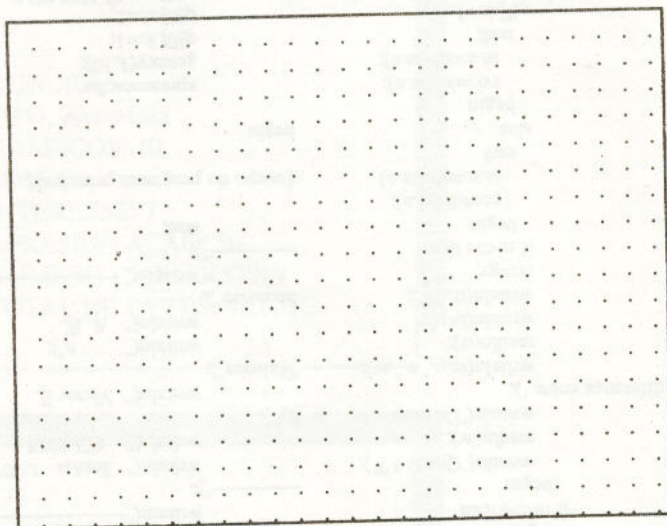
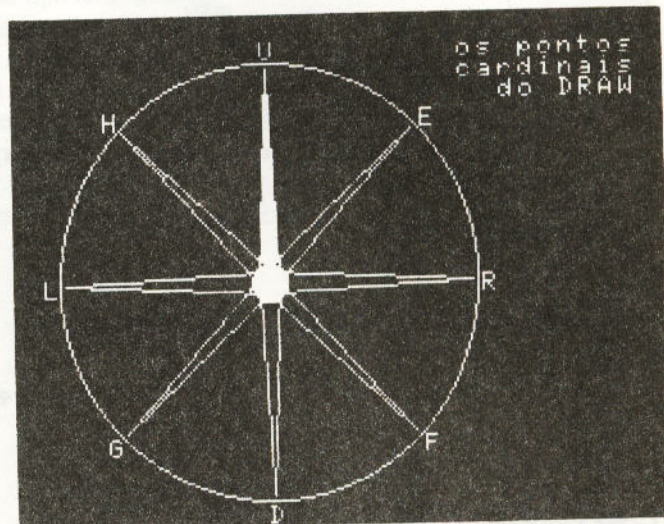
A montagem do desenho é feita através da string que deve conter uma sequência de códigos cuja memorização e compreensão é relativamente simples.

Para melhor entender o que será discutido a seguir, você deve ligar seu MSX e digitar o programa da figura 1.

FIG.1

```
10 SCREEN 2
20 LINE(0,0)-(255,191),,B
30 FOR L=0 TO 191 STEP 10
40 FOR C=0 TO 255 STEP 10
50 PSET (C,L)
60 NEXT C,L
1000 GOTO 1000
```

Rode-o e você terá uma retícula na SCREEN 2, que servirá como referência para os desenhos que iremos gerar (figura 2).

FIG.2

Você deve imaginar um ponto se movendo na tela, como se fosse a ponta de um lápis, traçando as figuras que você deseja. Visualizando a tela como se fosse um mapa, com o Norte para cima, este lápis pode ser movimentado em 8 direções diferentes (figura 3).

Para memorizar, basta lembrar que os 4 pontos cardinais principais são indicados pelas iniciais das direções em inglês:

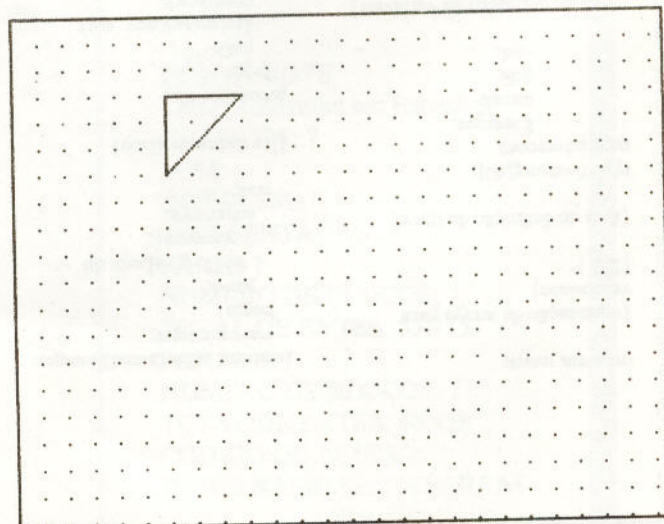
U = Up (para cima)
D = Down (para baixo)
L = Left (para esquerda)
R = Right (para direita)

e os secundários (diagonais) estão em ordem alfabética (E, F, G, H), começando do Norte e girando no sentido horário.

Breque o programa na tela com CONTROL+STOP, acrescente as linhas da figura 4 e rode-o. Você deve obter a figura 5.

FIG 4

```
100 DRAW"BM60,60"
110 A$="U30R30G30"
120 DRAW A$
```

FIG 5

A linha 100 localiza a ponta do "lápis" nas coordenadas (60,60), como você pode verificar pela retícula.

A linha 110 define uma variável string (A$) que contém a sequência de comandos a serem executados pela 120:

U30 = desenha uma linha de 30 pontos para cima (Up)

R30 = desenha uma linha de 30 pontos para direita (Right)

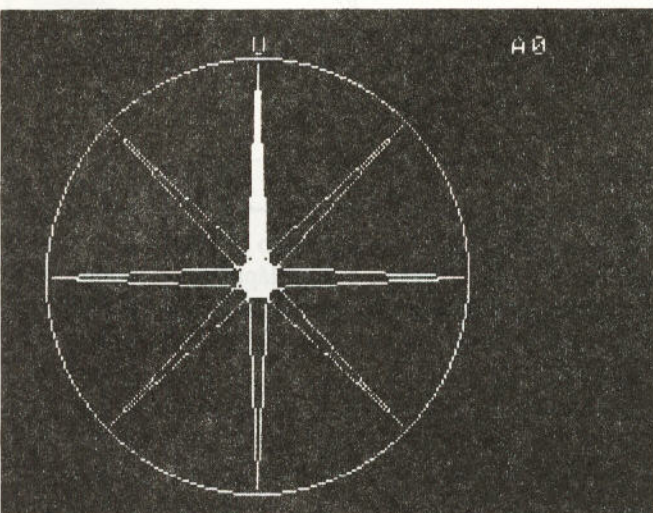
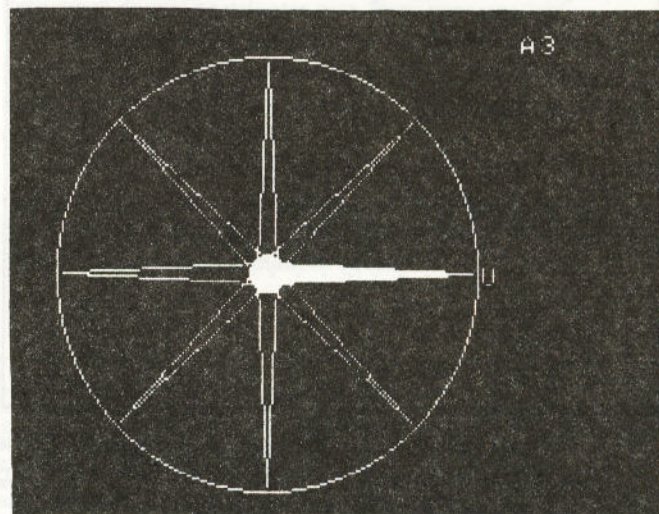
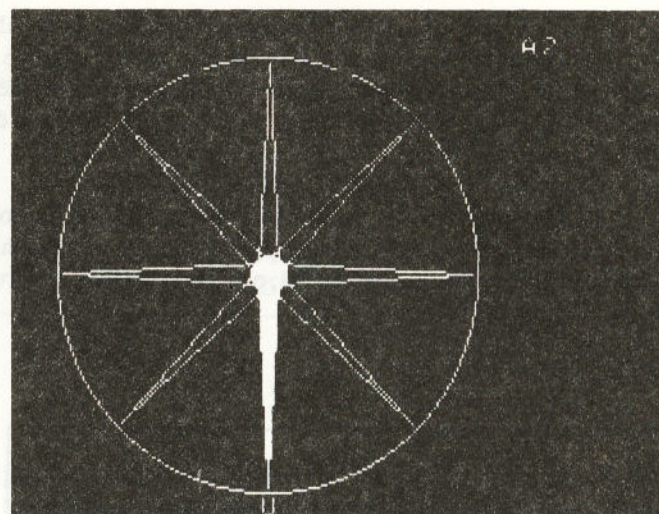
G30 = desenha uma linha de 30 pontos para baixo e 30 pontos para esquerda.

Agora que seu desenho foi definido pela variável A$, você pode fazer algumas alterações: a escala do desenho, por exemplo, pode ser alterada pelo parâmetro S. O valor inicial (default) é S4, pois cada unidade corresponde a 1/4 de ponto.

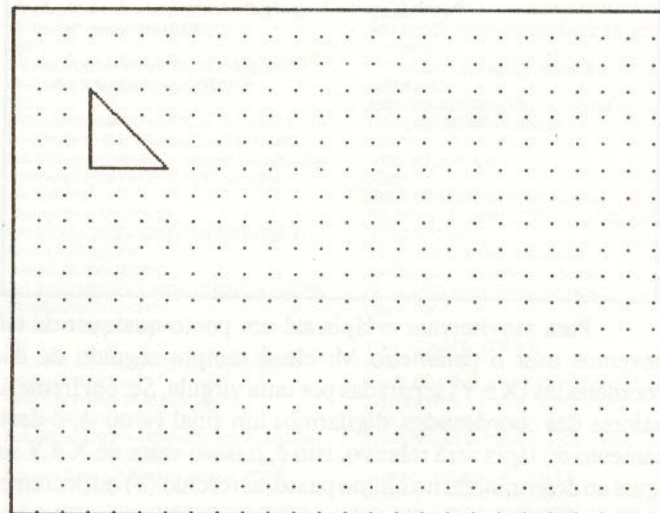
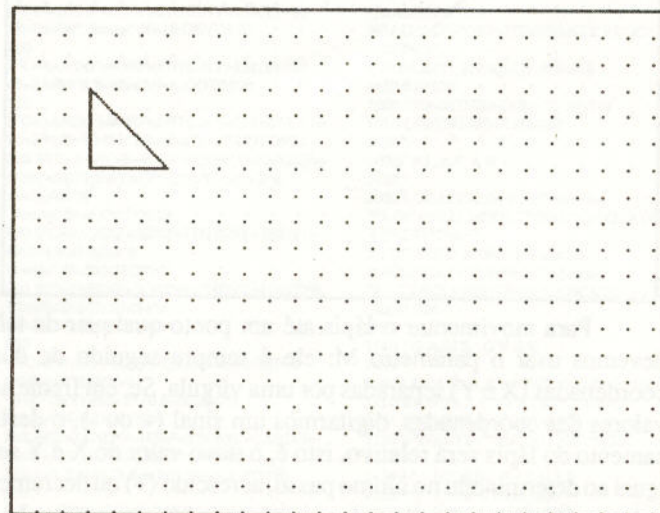
Experimente acrescentar ao seu programa a linha:
115 DRAW "S8"e você estará dobrando a escala do desenho.

Rode o programa assim alterado para se certificar do efeito.

Você pode, também, alterar a orientação dos seus "pontos cardinais" usando o parâmetro A. As orientações definidas por A estão detalhadas na figura 6.

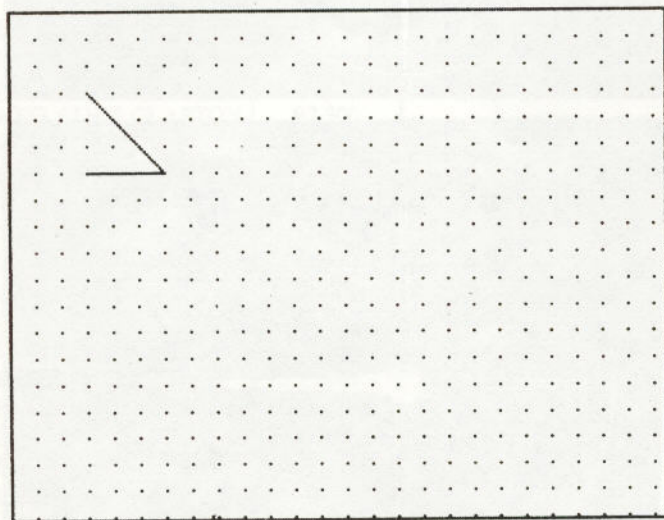
FIG 6

Experimente alterar a linha 115 para:
115 DRAW "S4A1"de maneira a reestabelecer a escala inicial (S4) e rodar a orientação dos eixos de 90 graus no sentido anti-horário (A1).

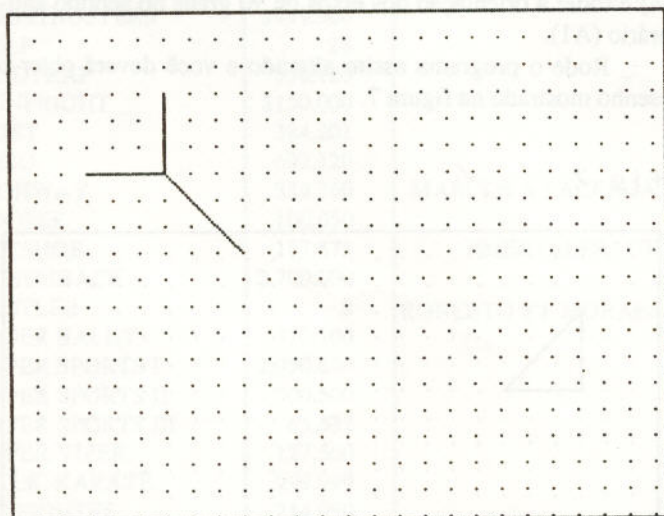
Rode o programa assim alterado e você deverá obter o desenho mostrado na figura 7.

fig 7

Vamos, agora, entender o parâmetro B, já utilizado na linha 100 do programa. Ele permite movimentar o "lápis" sem que o mesmo desenhe nada. Exemplificando melhor, leve o cursor até a linha 110 e, usando a tecla INSERT, coloque um B na frente do R30 de maneira a obter:

110 A$= "U30BR30G30"

Rode novamente o programa com esta alteração e note como o lápis se desloca de 30 pontos para direita sem desenhar nada. Cuidado, como estamos em A1, o nosso "para direita" virou "para cima" na tela (figura 8).

figura 8

Um outro parâmetro importante é o N: ele permite desenhar passos sucessivos fazendo o "lápis" voltar sempre à posição inicial. Altere, novamente, a linha 110 para:

110 A$ = "NU30NR30NG30"e rode o programa assim alterado. Você obterá o desenho da figura 9.

Para movimentar o lápis até um ponto qualquer da tela, devemos usar o parâmetro M: ele é sempre seguido de duas coordenadas (X e Y) separadas por uma vírgula. Se, em frente aos valores das coordenadas, digitarmos um sinal (+ ou -), o deslocamento do lápis será relativo, isto é, o novo valor de X e Y será igual ao determinado no ùltimo passo, acrescido (+) ou decrementado (-) do valor indicado pelas coordenadas que seguem o M.

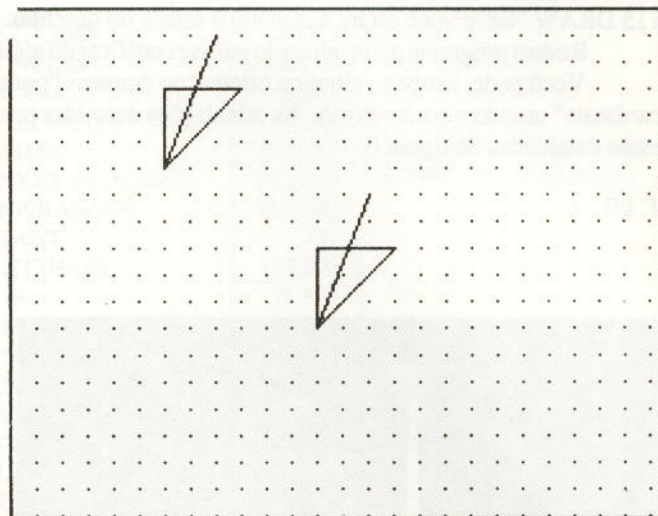
Para entender melhor, vamos fazer as seguintes alterações no nosso programa:

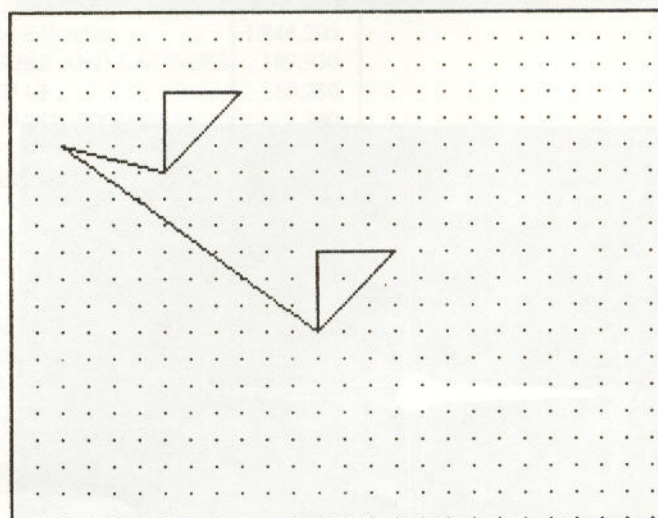
110 A$ = "U30R30G30M+20,-50"para voltar a desenhar nosso triângulo inicial, acrescido de um deslocamento relativo,

115 DRAW "S4A0"para restabelecer a escala inicial e a orientação normal (U para cima na tela)

130 DRAW "BM120,120"

140 DRAW A$e para deslocar o "lápis" para as coordenadas (120,120) e desenhar novamente a figura definida por A$.

Você deverá obter o desenho da figura 10.

Se as coordenadas que seguem o M não tiverem sinal, o deslocamento será absoluto, ou seja, o "lápis" se deslocará sempre para o mesmo ponto independentemente de sua posição no ùltimo passo. Experimente alterar a linha 110 retirando os sinais das coordenadas.

110 A$ = "U30R30G30M20,50"e rode o programa assim alterado. Você obterá o desenho da figura 11. Note que a posição final do "lápis" é a mesma em ambas as figuras!

O parâmetro C permite definir a cor do "lápis" (de 0 a 15).

Na SCREEN 2 deve-se tomar o cuidado de usar a mesma cor na horizontal de 8 em 8 pontos para não pintar inadvertidamente uma área vizinha. Isto significa que, para um dado Y, se usamos uma certa cor no X de 0 a 7, ou de 8 a 15, ou de 16 a 23, etc, não podemos definir outra cor no mesmo intervalo, sob pena de alterar a cor do que já foi desenhado.

A expressão string que define o desenho a ser executado pelo DRAW pode conter variáveis (numéricas ou alfanuméricas). Neste caso, elas devem ser precedidas por um "prefixo" e seguidas de um "sufixo" para que o DRAW posa reconhecê-las.

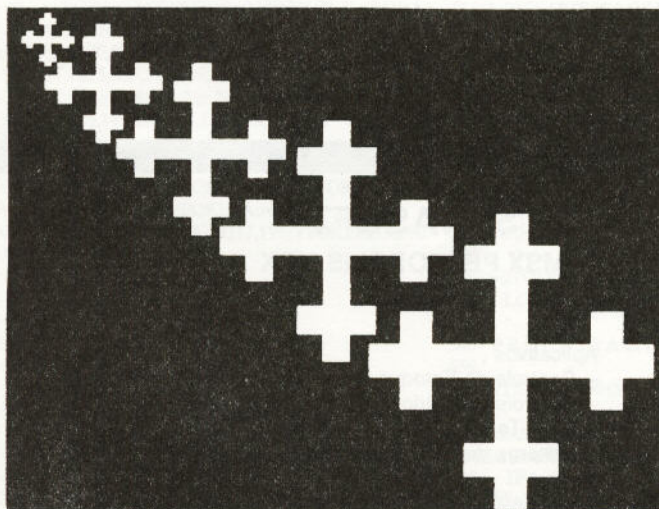
As variáveis numéricas devem ser precedidas por um "=" e seguidas por um ";". As variáveis string (alfanuméricas) devem ter um "x" à frente e também devem ser seguias por um ";".

Vamos fazer uma experiência para compreender melhor a utilidade deste recurso. Apague ou grave o programa que estamos usando e digite o da figura 12.

FIG 12

```
100 SCREEN 2
110 LINE(0,0)-(255,191),,B
120 A$="U20R10U10L10U10L10D10L10D10R10D
20"
130 FOR E=5 TO 1 STEP -1
140 PX=7.7*(E^2)+10:PY=4.8*(E^2)+8
150 DRAW"BM=PX;,=PY;"
160 FOR D=0 TO 3
170 DRAW"S=E;A=D;XA$;"
180 NEXT D
190 PAINT(PX-1,PY+1)
200 NEXT E
210 GOTO 210
```

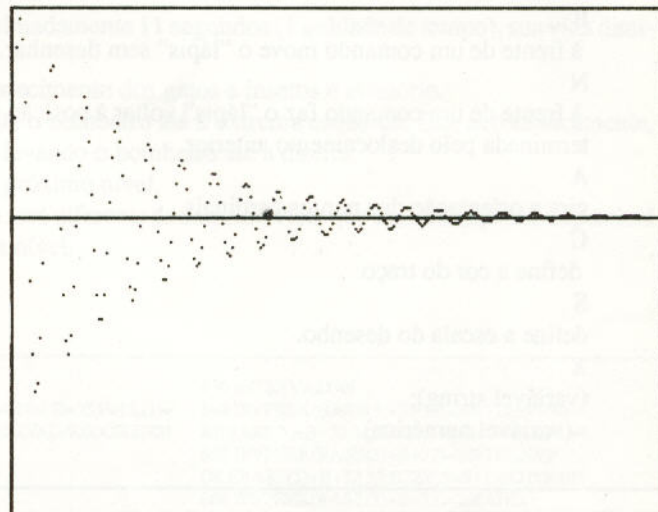
Ao rodá-lo, você deve obter o desenho mostrado na figura 13.

FIG 13

Você deve ter notado a utilidade deste recurso (inclusão de variáveis na string do DRAW) especialmente na construção de gráficos de funções matemáticas. Ao invés de usar o PSET e obter uma sucessão de pontos, você pode usar o DRAW e obter uma curva contínua.

Digite, por exemplo, o programa listado na figura 14 para obter o gráfico de oscilações amortecidas em função do tempo (figura 15).

FIG 14

```
100 SCREEN 2
110 LINE(0,0)-(255,191),,B
120 LINE(0,80)-(255,80)
130 FOR X=0 TO 250
140 Y= 80*(EXP(-.02*X))*SIN(.5*X)
150 YC= 80-Y
160 PSET(X,YC)
170 NEXT X
180 GOTO 180
```

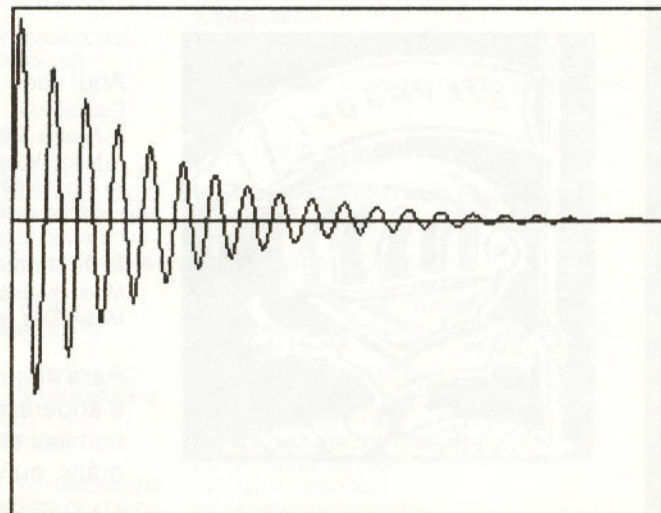
FIG 15

Como você pode notar, a primeira parte do gráfico é extremamente confusa e só pode ser melhorada se colocarmos um STEP bem pequeno no laço do X, tornando o programa lentíssimo.

Altere, agora, as seguintes linhas:

100 SCREEN 2:DRAW "BM0,80"

160 DRAW "M=X;,=YC;"e veja o efeito na figura 16.

Melhorou bastante, não?

FIG 16

Resumindo, então, tudo o que aprendemos:

U, D, R, L, E, F, G, H movem o "lápis" nas direções dos pontos cardinais.
M(+-X,+-Y) move o "lápis" desde o último ponto até as coordenadas indicadas.

Se as coordenadas não tiverem sinal à sua frente, o deslocamento é absoluto. Se tiverem sinal, o deslocamento é relativo e os valores de +-X e +-Y indicam acréscimos ou decréscimos em relação à última posição.

B
à frente de um comando move o "lápis" sem desenhar.
N
à frente de um comando faz o "lápis" voltar à posição determinada pelo deslocamento anterior.
A
gira a orientação dos pontos cardinais.
C
define a cor do traço.
S
define a escala do desenho.
X
(variável string);
=(variável numérica)

O Prof. Pierluigi Piazzi é autor de vários livros para MSX dentre os quais o "CURSO DE BASIC MSX vol.2", a ser publicado brevemente pela Editora Aleph, a partir do qual foi adaptado este texto.

ARQUIVOS DE LOTE

Descubra esta facilidade do DOS

BRUNO MARRUT

Você já deve ter tido a necessidade de executar uma sequência de comandos várias vezes no DOS.

Para facilitar esta tarefa, existe um tipo de arquivo, o arquivo de lote, que é um arquivo de texto ASCII, podendo, portanto, ser criado em qualquer processador de texto que efetue a gravação neste formato, devendo ter, necessariamente, a extensão ".BAT"

Um arquivo de lote também pode ser utilizado com o comando COPY do DOS, utilizando-se a seguinte sintaxe: COPY CON xxx.BAT (xxx representa o nome do arquivo a ser criado).

Após o comando COPY, usamos o dispositivo CON, que vem a significar teclado, ou seja, os dados digitados via teclado serão gravados no arquivo xxx.BAT. Para finalizarmos a operação, indicarmos ao computador que desejamos parar de gravar informações no arquivo e que ele pode encerrar a operação, devemos teclar ^Z. Com este comando, o disk drive deve ser acionado e o DOS gravará o arquivo xxx.BAT.

O comando COPY possui, além da função mais básica e que é por todos utilizada, que é a de copiar arquivos de um disco para outro, a de gerar arquivos de lote, conforme foi visto acima, e a de copiar um arquivo que se encontre em ASCII na impressora, à medida que o mesmo é visualizado na tela. Assim, um texto gravado através do MSXWRITE, por exemplo, pode ser impresso com o comando COPY. Como exemplo, para um arquivo gravado em ASCII de nome CPU.TXT

COPY CPU.TXT PRN

Logicamente, se o arquivo não estiver no formato ASCII, a operação acima resultará em alguma coisa de estranho.

Mas a função deste artigo não é explicar aos leitores o comando COPY, e sim as vantagens do arquivo de lote.

Nos arquivos de lote devem ser utilizados comandos do DOS, da mesma forma que os mesmos seriam digitados via teclado, sendo que cada arquivo poderá conter um ou mais comandos, desde que cada comando esteja em uma linha separada.

A execução de um arquivo lote se dá da mesma forma que um comando do DOS, ou seja, digitando-se o seu nome, sendo que não há necessidade de se informar a extensão ".BAT".

Poderemos criar, por exemplo, um arquivo de lote que nos forneça o diretório do disco. A este arquivo podemos chamar simplesmente de D.BAT. Assim, ao digitarmos D no DOS, o mesmo processará o comando DIR de acordo com os parâmetros que especificamos no arquivo de lote. Concluindo, economizamos o tempo de digitação de vários caracteres.

Exemplo:

```
COPY CON D.BAT     + RETURN
DIR                + RETURN
^Z
```

Digite:

```
D + RETURN
```

Você deverá obter como resultado o diretório do disco do drive corrente.

Mas os arquivos de lote possibilitam muito mais do que o prático exemplo acima.

Vamos supor que você trabalhe muito com o dBase ou com o Supercalc ou com qualquer programa que gere arquivos de dados que devem ser sempre copiados por questão de segurança. Nestes casos, podemos construir um arquivo de lote que chame o programa e que, após termos deixado de utilizá-lo, faça um backup do disco ou dos arquivos necessários, eliminando, assim, aquele fator preguiça que, ás vezes, nos sai tão caro.

É importante observamos que quando chamamos um programa dentro de um arquivo de lote, desde que o mesmo possa ser carregado no Sistema Operacional, quando encerrarmos a execução do mesmo, o controle voltará ao arquivo de lote em execução e não ao Sistema Operacional, o que possibilita efetuarmos o backup descrito acima. Da mesma forma, se dentro de um arquivo de lote fizemos a chamada de um outro arquivo de lote, o controle passará a ser deste e não daquele que o chamou. Ou seja, se em um arquivo de lote fizermos a chamada de um outro arquivo de lote, de nada adianta colocarmos comandos para serem executados depois desta chamada, pois os mesmos não serão executados.

Certos comandos do DOS podem fazer um grande estrago nos discos, caso não sejam bem utilizados, como é o caso do FORMAT, do COPY e do ERASE. Estes comandos também podem ser incluídos em um arquivo de lote a fim de tornar mais segura a operação de cópia de arquivos, principalmente nos casos onde se utilizam dados que, ao fim de uma seção de trabalho, devem ser copiados no arquivo de extensão ".ANT", cujos dados, por sua vez, devem ser copiados em outro arquivo cuja extensão é ".VEL", por exemplo.

Um arquivo de lote poderá fazer estas tarefas perfeitamente sem que nos arrisquemos que uma falha nossa possa prejudicar todo o trabalho.

Caso o nosso arquivo de lote tenha o nome de AUTOEXEC.BAT, este será executado automaticamente pelo DOS. Existem várias aplicações para este tipo de arquivo, como por exemplo chamar automaticamente um programa que se encontra em um disco, mudar a cor de fundo da tela e voltar para o sistema, etc.

Exemplo: mudança da cor de fundo

O programa AUTOEXEC.BAT poderá ser o seguinte:

```
COPY CON AUTOEXEC.BAT + RETURN
BASIC COR.BAS
^Z
```

O programa em BASIC poderá ser este.

```
10 COLOR 15,1:CALL SYSTEM
```

No exemplo acima utilizamos o programa AUTOEXEC.BAT para chamarmos o BASIC e rodar o programa COR.BAS, que também tem que estar gravado no mesmo disco, logicamente. O programa em BASIC efetua a mudança da cor de fundo e chama novamente o sistema.

Já existem rotinas que permitem efetuar esta operação diretamente do Sistema Operacional e que serão vistas em outros artigos.

Este exemplo pode parecer um tanto trabalhoso na primeira vez que for executado, mas, depois, a cor de fundo da tela será mudada automaticamente, sem que o usuário tenha que mexer um único dedo sequer.

Existem comandos do DOS cuja função é facilitar o processamento dos arquivos de lote, como o REM e o PAUSE.

O comando PAUSE tem a finalidade de parar o processamento até que o usuário decida continuar, pressionando uma tecla. Este comando pode ser utilizado, também, com a função de solicitar ao usuário uma confirmação para a realização do próximo passo.

Podemos utilizar REM para imprimir informações na tela a respeito do que está sendo processado. Portanto, REM e PAUSE são usados, geralmente, em conjunto e permitem ao usuário um total controle da situação.

Assim, poderíamos incrementar, de certa forma, o exemplo dado acima para efetuarmos a mudança de cor.

```
COPY CON AUTOEXEC.BAT
                    + RETURN
REM INSIRA DISCO COM O PROGRAMA
COLOR.BAS           + RETURN
PAUSE               + RETURN
BASIC COLOR.BAS     + RETURN
^Z
```

O programa COLOR.BAS não necessita de alteração.

Neste caso, caso houvessem comandos para serem executados após a linha BASIC COLOR.BAS, os mesmos não seriam executados, pois efetuamos a chamada do BASIC. Para voltarmos ao Sistema Operacional, basta darmos o comando CALL SYSTEM, pois, ao inicializarmos, efetuamos o carregamento do mesmo.

Os arquivos de lote podem ter o seu processamento interrompido ao se pressionar as teclas CTRL STOP, sendo que o sistema irá solicitar confirmação para o encerramento do trabalho.

Neste artigo lhe demos as condições para que você possa utilizar este poderoso comando do DOS e que somente em poucos manuais que acompanham os Drives vem bem explicado

As utilizações que você irá dar caberão exclusivamente às suas necessidades.

SCREEN 0

Tirando todas as vantagens da screen 0

ANDRÉ L. DE FREITAS

O processador de vídeo dos micros da linha MSX possui vários modos de tela, conforme já é de conhecimento dos usuários de, visto que o comando SCREEN, do BASIC, permite a entrada em quatro tipos de tela diferentes.

Todas estas telas possuem uma estrutura diferente e utilizam diferentes áreas da memória de vídeo para trabalho.

Neste artigo, veremos como está estruturada a memória de vídeo na SCREEN 0 e como poderemos tirar proveito de algumas rotinas existentes na ROM do micro para trabalhar, mesmo em BASIC, de forma rápida e elegante neste modo. Também veremos como usar estas rotinas em Assembly, caso você necessite delas em seus programas.

Não reparem se alguns nomes para variáveis ou labels que eu apresento sejam sem sentido para usuários não conhecedores do inglês, mas isto se deve ao fato de terem origem nesta língua.

Prefiro usar os originais, pois são os adotados por toda literatura mundial a respeito dos micros MSX.

Os exemplos fornecidos em mnemônicos do Z80 estão de forma a ser corretamente assemblados pelo assembler GEN 80, da Hisoft, em qualquer versão, disco ou fita.

Caso o seu assembler não seja o mesmo e apresente alguns erros, estes poderão ser por tamanho máximo de caracteres permitidos em Labels (geralmente 5 ou 6) ou, então, pela notação usada para designar constantes hexadecimais.

No GEN, usamos o símbolo # antes da constante que, por exemplo, supondo a constante #F000, poderá mudar para 0F000H ou somente &F000, dependendo do assemblador.

Nas rotinas apresentadas não estão definidos endereços onde as mesma serão alocadas pela pseudo-instrução ORG. Isto você irá definir de acordo com suas necessidades.

A ESTRUTURA VRAM NA SCREEN 0

O modo de vídeo correspondente ao SCREEN 0 é um modo de texto que apresenta 24 linhas de 40 caracteres cada uma, podendo haver somente uma cor para todos os caracteres no vídeo e uma só cor para o fundo de tela, não possuindo a característica de trabalhar com SPRITES.

Este modo está estruturado na VRAM (memória de vídeo) da seguinte forma:

Endereços

0 a 959 - área de caracteres na tela (24 x 40 = 960)

Endereços

2048 a 4095 - tabela de formação de caracteres, sendo que cada um dos 256 caracteres precisa de 8 bytes para sua definição (256 x 8 = 2048).

Para entender o que se passa, experimente o programa:

```
10   SCREEN 0: KEY OFF: CLS: J=0
20   FOR I = 0 TO 959
30   VPOKE I,J
40   J = (J + 1) MOD 256
50   NEXT I
```

Este programa mostrará como estão as posições da tela, colocando caracteres nas suas respectivas posições, escrevendo diretamente na VRAM.

As cores não estão mapeadas na VRAM, mas sim na memória RAM do micro, sendo transferidas diretamente ao processador de vídeo quando definidas pelo comando COLOR.

INICIALIZANDO A TELA DE TEXTO 40 x 24

No BASIC MSX, para acessarmos a SCREEN 0, simplesmente damos o comando SCREEN 0 (mais simples impossível).

Em Assembly, devemos fazer uma chamada à rotina INITXT, presente na ROM do BIOS no endereço &H6C.

Esta rotina tem como parâmetros as variáveis de memória TXTNAM (&HF3B3) e TXTCGP (&HF3B7).

TXTNAM guarda o endereço da memória de vídeo onde está a tabela de posições onde os caracteres aparecem na tela, que vamos chamar de tabela de posição de caracteres.

TXTCGP guarda o endereço da VRAM da tabela a partir da qual estão os bytes que definem a forma de cada caracter, oito bytes para cada caracter, a qual chamaremos tabela de definição dos caracteres.

Estas variáveis se encontram na RAM utilizada pelo BASIC e BIOS do MSX e devem possuir os valores 0 para TXTNAM e 2048 para TXTCGP.

Em Assembly, a chamada à INITXT se pareceria com o descrito abaixo:

```
INITXT:     EQU #006C;
              Rotina do BIOS
TXTNAM:     EQU #F3B3; var.
              Tabela de posições
TXTCGP:     EQU #F3B7; var.
              Tabela de caract.
INICTX:     LD HL,000
              posição na VRAM
            LD (TXTNAM),HL ;
              escreve na variável
            LD HL,#800;
              posição na VRAM
            LD (TXTCGP),HL;
              escreve na variável
            CALL INITXT;
              chama rotina
            RET;        fim
```

Isto equivale a você digitar, no BASIC, o comando SCREEN 0, sendo que em Assembly você deve tomar o cuidado de salvar os registradores do Z80 antes de chamar esta rotina e restaurá-los após a chamada para não criar confusões com outras rotinas que possam estar sendo usadas por seu programa. Esta rotina altera todos os registradores do Z80.

LIMPANDO A TELA

Quando queremos limpar a tela, no BASIC, simplesmente digitamos CLS. A rotina correspondente no BIOS se chama também CLS e é encontrada no endereço &HC3.

Para limpar a tela em Assembly, basta definir no seu programa um label chamado CLS com o endereço, na forma:

```
CLS:     EQU #00C3
```

Para limpar a tela, a qualquer hora, no seu programa, basta chamar esta rotina:

```
CALL CLS
```

MUDANDO AS CORES

As cores para caracteres e fundo não são definidas na VRAM, mas sim em duas variáveis de memória presentes na RAM do micro que é utilizada pelo BASIC.

As variáveis e seus endereços na memória são as seguintes:

FORCLR - cor dos caracteres
Endereço &HF3E9
BAKCLR - cor de fundo
Endereço &HF3EA

Estas variáveis são lidas pelo sistema MSX durante algumas operações que este realiza com o vídeo e são passadas pelo micro para um determinado registrador do processador de vídeo, sendo que, se quisermos alterar as cores, devemos escrever seu código nestas variáveis e depois chamar a rotina para alterá-las. É desta forma que, inconscientemente, estamos agindo, pois o micro faz todo o trabalho quando damos um comando como COLOR 15,1 no BASIC.

Se quisermos esta operação realizada sem o comando COLOR, devemos setar as variáveis de memória com os valores das cores correspondentes e depois chamar a rotina existente na ROM que faça esta alteração, que é chamada de CHGCLR e se encontra no endereço &H62.

Isto pode ser feito da seguinte forma:

```
10 DEFUSR = &H62           REM
   Rotina CHGCLR
20 POKE &HF3E9,8           REM
   Cor de caracteres   (FORCLR)
30 POKE &HF3EA,4           REM
   Cor de fundo        (BAKCLR)
40 X = USR(0)
```

É claro que usar um programa destes para mudar as cores no BASIC é desperdício e nada elegante, mas, em Assembly, ele funciona de forma semelhante. Por isso eu mostrei o equivalente em BASIC.

Em Assembly, seria feito da seguinte forma:

```
CHGCLR:    EQU #0062
           ; Rotina da ROM
FORCLR:    EQU #F3E9
           ; var. cor de caraçt.
BAKCLR:    EQU #F3EA
           ; var. cor de fundo
MUDCOR:    LD A,8 ;
           cor de caracteres
```

POSICIONANDO O CURSOR E IMPRIMINDO CARACTERES

Para posicionar o cursor e imprimir um caracter, em BASIC, procederíamos como abaixo:

```
10 LOCATE 10,10
20 PRINT "A"
```

Se fosse um caracter especial, o caracter de código 1, por exemplo, poderíamos fazer:

```
10 LOCATE 10,10
20 PRINT CHR$(1)
```

Se quiséssemos escrever direto na VRÁM, teríamos que calcular o endereço:

```
10 LINHA= 10
20 COLUNA= 10
30 CALC=40 * LINHA + COLUNA
40 VPOKE CALC,1
```

Em linguagem de máquina, temos uma rotina no BIOS que imprime um caracter na posição corrente do vídeo. Esta é uma forma de se fazer a operação, desde que, antes, alteremos a posição corrente de impressão. Alterar esta posição é facil, pois existem duas variáveis de sistema: a CSRX e a CSRY que contêm a abscissa e a ordenada da posição na tela.

CSRY está no endereço &HF3DC e CSRX em &HF3DD na RAM. Se escrevermos em CSRY a nossa linha e em CSRX a coluna em que se quer posicionar, na próxima vez que imprimirmos um caracter, o mesmo estará nesta posição. Outro modo de posicionar é escrever no registrador H a coluna, no registrador L a linha e chamar a rotina POSIT (&HC6) do BIOS.

Para posicionamento via BIOS, teremos que considerar as posições válidas como entre 1 e 40 para coluna e 1 e 24 para linha, ao contrário do 0 e 39 para coluna e 0 e 23 para linha usado no BASIC.

Ainda poderíamos calcular, à parte, o endereço da memória de vídeo onde se quer apresentar o caracter e lá escrever diretamente um byte correspondente, mas isto daria muito trabalho. Então, por enquanto, vamos deixar de lado este método, pois envolve acesso ao processador de vídeo via portas lógicas do micro.

Vou mostrar, a seguir, os dois métodos descritos acima.

Método de posicionamento por variável de de memória:

```
CSRY:  EQU #F3DC
CSRX:  EQU #F3DD
POSIC: LD A,10        ; coluna
       LD (CSRX),A
       LD A,5         ; linha
       LD (CSRY),A
```

ou então, simplificando:

```
CSRY:  EQU #F3DC
POSIC: LD HL,#0A05    ; 0A05
       (0A = coluna 10; e 05 = linha 5)
       LD (CSRY),HL
       ; escreve os dois bytes
       ; ja que CSRY e CSRX
       ; são subsequentes.
```

Método pela rotina POSIT:

```
POSIT: EQU #00C6
POSIC: LD H,10; coluna
       LD L,5 ; linha
       CALL POSIT
```

OBS: A rotina POSIT modifica o valor do registrador A.

Para imprimir o caracter, usaremos a rotina CHPUT no endereço &HA2, cujo caracter a ser impresso deve estar no registrador A, antes da chamada. Como exemplo:

```
CHPUT:EQU #00A2
IMPRIM:        LD A,65; caracter "A", cod. 65
CALL CHPUT
```

Com as rotinas acima juntas, podemos posicionar e escrever um caracter de forma simples.

IMPRESSÃO DE STRINGS

Para a impressão de uma string, basta um Loop em linguagem de máquina que imprima do primeiro ao último caracter, usando o byte 0, por exemplo, para indicar o fim da string.

Para posicionar, basta usar a rotina de posicionamento uma só vez antes de começar a imprimir.

Como exemplo, o programa a seguir imprime:

REVISTA CPU - MSX"

```
CHPUT:   EQU #00A2
IMPSTR:  LD HL,STRING
         ; apontaHL p/ string
LOOP:    LD A,(HL)
         ; A recebe caracter
CP 0     ; testa se é fim
JR Z,FIM ; vai p/ fim se zero
CALL CHPUT ; impr. caracter
INC HL   ; avança ponteiro
JR LOOP  ; volta
FIM:     RET
STRING:  DEFM
"REVISTA CPU - MSX"
DEFB 0   ; marca de fim de string
```

COMO ELIMINAR A LINHA DE FUNÇÕES

Existem duas rotinas no BIOS: uma delas retira do vídeo a linha de funções e a outra repõe a mesma em seu devido lugar.

São elas:

ERAFNK (&HCC) - apaga funções
DSPFNK (&HCF) - mostra funções

A maneira pela qual devem ser utilizadas é semelhante à rotina de CLS. Estas rotinas modificam todos os registradores.

ESCREVENDO DIRETO NA VRAM

Existem algumas rotinas no BIOS que executam a mesma função que, por exemplo, os comandos VPEEK e VPOKE do BASIC, e até mesmo copiam blocos inteiros de bytes da RAM para a VRAM e vice-versa.

Estas rotinas funcionam na SCREEN 0, desde que respeitados os endereços das posições ou da tabela de caracteres neste modo.

São estas as rotinas do BIOS:

RDVRM (&H4A) -

lê um byte da VRAM, no endereço especificado por HL, retornando o byte no registrador A.

WRTVRM (&H4D) -

escreve um byte na VRAM, no endereço especificado por HL, sendo este byte passado pelo registrador A.

FILVRM (&H56) -

preenche uma área da VRAM com o byte especificado pelo reg. A, começando no endereço especificado pelo registrador HL, sendo a quantidade de bytes passada pelo registrador BC.

LDIRVM (&H5C) -

copia um bloco de bytes de tamanho especificado por BC, da área da RAM apontada por HL para a área da VRAM apontada por DE.

LDIRMV (&H59) -

copia um bloco de bytes de tamanho especificado por BC, da área da VRAM apontada por HL para a área da RAM apontada por DE.

As duas primeiras rotinas somente modificam o conteúdo do registrador A, e as demais alteram o conteúdo de todos os registradores.

Como exemplo de utilização, o programa abaixo enche a tela com caracteres "A":

```
FILVRM: EQU #0056
ENCHE:  LDHL,0 ; inicio da        VRAM
        LD BC,960; (40 x 24)
        caracteres
        LD A,65       ; caracter "A"
        CALL FILVRM
        RET
```

ROTINAS ADICIONAIS

Estas funções que vou descrever a seguir não têm relação direta com a SCREEN 0, mas são ùteis quando se tenta desenvolver um programa que dependa de informações apresentadas no vídeo.

Desabilitando e habilitando o processador de vídeo:

Usado quando se quer montar uma tela e depois torná-la visível de uma só vez, já pronta. Crie o seu programa, montando a tela normalmente e, antes de começar o código que gera a tela, inclua uma chamada à rotina DISSCR (&H41). A imagem que o processador de vídeo envia para o circuito do monitor some, ficando desabilitada, mas o processador continua a operar de forma correta e normal. Quando terminar a montagem da tela, habilite novamente o vídeo com a rotina ENASCR (&H44). A tela surgirá toda de uma vez, já pronta.

Experimente:

```
10 DEFUSR =&H41:REM DISSCR
20 DEFUSR1=&H44:REM ENASCR
30 CLS
40 X =USR(0)
50 FOR I =0 TO 14
60 LOCATE I,I : PRINT "TELA MON
              TADA  SEM VOCE VER"
70 NEXT I
80 X =USR1(0)
```

Lendo um caracter do teclado e imprimindo numa posição do vídeo:

A rotina CHGET (&H9F) realiza a leitura de um caracter do teclado, não se referindo a SCREEN 0, mas muitas vezes, num programa com seleção de opções apresentadas no vídeo, desejamos ler uma opção e escrever ao lado de uma pergunta apresentada na tela. Podemos fazer, utilizando as rotinas de posicionamento e impressão de strings já descritas, o seguinte:

```
CHGET:  EQU #009F
CHPUT:  EQU #00A2
CSRY:   EQU #F3DC
OPCAO:  LD HL,#0A05; posiciona
        LD (CSRY),HL
        LD HL,MESG;
                aponta mensagem
        CALL LOOP; imprime
        CALL CHGET; lê caracter e re
torna; o mesmo no reg. A
        CALL CHPUT; impr. caracter
no reg. A
        RET
LOOP:   LD A,(HL)A recebe caracter
        CP 0    ; testa se e fi
        JR Z,FIM; vai p/ fim se zero
        CALL CHPUT  ; impr. caracter
        INC HL; avança ponteiro
        JR LOOP; volta
FIM:    RET
MESG:   DEFM "Qual a opção ? "
        DEFB 0
```

PAGINANDO A SCREEN 0

Um truque pode ser feito na SCREEN 0 e aproveitar o resto do espaço livre da VRAM não utilizado pelo processador para simular páginas de vídeo, montando telas em cada uma e "chaveando" estas telas como quiser. Existe a função BASE, no BASIC MSX e esta função trabalha com as características dos modos de vídeo.

O BASE(0) guarda o endereço inicial da tabela de posições na tela da SCREEN 0. Mudando o valor do BASE(0) que, normalmente, é zero, podemos fazê-lo apontar para outras áreas da VRAM.

Por uma certa característica de endereçamento do processador de vídeo, devemos alterar o BASE(0) somente em múltiplos de 1024 bytes e não podemos fazê-lo coincidir com o endereço da tabela de caracteres, senão não entenderemos mais nada do que estiver no vídeo, pois estaremos escrevendo no vídeo e por cima da tabela de caracteres simultaneamente.

O programa em BASIC, abaixo, vai montar telas nas "páginas" 1 e 2 escolhidas arbitrariamente e ficará alternando as mesmas.

```
100 ON STOP GOSUB 220: STOP ON
110 BASE(0) = 1024: SCREEN 0
120 LOCATE 10,10: PRINT "TELA EM
      PAGINA 1"
130 BASE(0) = 4096: SCREEN 0
140 LOCATE 8,8 : PRINT "
150 LOCATE 8,10: PRINT "* TELA EM
      PAGINA 2
160 LOCATE 8,12: PRINT "
170 FOR I=1 TO 500: NEXT I
180 BASE(0) = 4096
190 FOR I=1 TO 500: NEXT I
200 BASE(0) =1024
210 GOTO 170
220 BASE(0) = 0 : SCREEN 0
230 RETURN 240
240 END
```

O programa acima simula a presença de páginas de memória no MSX.

Sempre que mudamos o valor de BASE(0), estamos alterando o ponteiro de onde se encontra a tabela de posições no vídeo presente na VRAM.

A cada comando SCREEN 0 colocado após a mudança do valor de BASE, inicializamos o modo zero para este novo endereço do ponteiro, mas não destruindo o conteùdo de outras partes da VRAM.

Depois de cada mudança, escrevemos o que quisermos no vídeo e tudo ficará guardado em áreas da VRAM diferentes.

A cada nova mudança, agora, do valor de BASE, voltamos a apontar para estas regiões "salvas" na memória de vídeo, fazendo os dados presentes aí reaparecerem na tela.

Para simplesmente apresentarmos os dados guardados não precisamos da instrução SCREEN. Caso contrário, estaremos alterando novamente a VRAM, mas somente do valor de BASE correspondente à área que se deseja apresentar.

CONCLUSÃO

Vimos, no transcorrer deste artigo, diversas rotinas referentes ao modo 0 de vídeo.

De posse destas rotinas, podemos gerar telas de texto, formatá-las da forma desejada, obtendo recursos em Assembly, da mesma forma que em BASIC.

O próprio interpretador BASIC dos micros MSX usa estas rotinas para o acesso ao vídeo. Portanto, todas são confiáveis.

Quando quisermos criar um programa em Assembly, é muito provável a existência de telas de texto neste.

Teremos, aí, uso para todas as rotinas descritas, de acordo com nossas necessidades.

Mesmo que nosso programa seja em BASIC, podemos agilizar certas partes, utilizando o Assembly, definindo então endereços de execução da parte em linguagem de máquina através das funções DEFUSR e USR do BASIC, e melhorando em muito nosso Software.

Espero ter agradado com esta descrição (talvez não totalmente detalhada) e seria demais querer aprofundar o conhecimento nos fundamentos destas rotinas, pois o objetivo foi tornar de fácil acesso a todos os níveis de usuários, mesmo que possuam somente um pequeno conhecimento de linguagem de máquina, coisas que existem no micro MSX e talvez você nao conhecesse.

Para quem quiser se aventurar, tente, com um desassembler, observar como estas rotinas foram escritas na ROM.

PLOTADOR GRÁFICO

Veja como traçar gráficos bidimensionais de uma função de uma variável.

ANTONIO F. SHALDERS

A finalidade desta série de artigos é mostrar ao leitor algumas das possíveis aplicações do MSX na área científica.

Embora muito pouco difundido entre os usuários da linha MSX, o plota-dor gráfico é de grande utilidade para quem lida com matemática e é especialmente indicado para estudantes universitários.

O primeiro programa da série é apresentado neste n´mero e é de aparência um tanto peculiar, principalmente o jeito como estão escritas as instruções, usando e abusando das zonas de tabulação e da função SPC.

Uma grande inovação é o uso de um cursor que permite a leitura do valor da função em um ponto específico. É uma aplicação séria para os sprites. Eles não estão aí somente para joguinhos ou para efeito decorativo !

DESCRIÇÃO

Este programa traça gráficos bidimensionais de uma função (função de uma variável). O ajuste de eixos é feito automaticamente.

Como era de se esperar, a execução do programa é um tanto lenta, pois o BASIC não é uma linguagem adequada para cálculos, mas a lentidão é perfeitamente suportável. Uma das principais causas dessa baixa velocidade é que o gráfico é composto por centenas de segmentos de retas, o que melhora muito o aspecto final do mesmo.

Foram utilizadas variáveis de simples precisão e inteiras a fim de aumentar a velocidade, pois, se usássemos dupla precisão, o programa iria tornar-se, pelo menos, 50% mais lento.

O programa é auto explicativo e de facílima utilização.

Para acessarmos o modo de análise da curva, basta pressionarmos a barra de espaço quando o gráfico estiver pronto, e RETURN para continuar.

No próximo artigo da série será apresentado um outro tipo de plotador.

Durante a continuação desta série, serão mostrados programas em outras linguagens com filosofia de construção e operação extremamente interessantes.

```
100 SCREEN 0:WIDTH 40:KEY OFF
110 ON ERROR GOTO 820
120 PRINT"PLOTTER ver. 1.0","(C) 1988 by CPU","A.F. Shalders & V. Welman","______________________":LOCATE 0,15
130 POKE &HFCAB,1
140 PRINT"Deseja instruções (S/N) ? ";
150 A$=INPUT$(1):IF A$="S" THEN 160 ELSE IF A$="N" THEN 190 ELSE 150
160 CLS:PRINT"INSTRUÇÕES DE USO:",,,"ATENÇÃO: PARA OPERAR O PROGRAMA CORRETA";SPC(9);"MENTE É NECESSÁRIO UM CERTO CO";SPC(9);"NHECIMENTO DE BASIC QUANTO À";SPC(9);"EDIÇÃO DE LINHAS.",,,
170 PRINTSPC(5);"A função a ser plotada fica residente no programa, sendo que para troca-la,é necessário alterar a mesma,que está nalinha 220, pela desejada.",,,SPC(5);"Responda a pergunta relativa a isto e só então o programa continuará a execução.",,,,
180 PRINTSPC(5);"Se for desejada a alteração , oprograma irá listar a linha que contém afunção. Modifique-a e pressione [F1] eo programa irá prosseguir normalmente.",,,"> Pressione uma tecla:";:A$=INPUT$(1)
190 KEY1,CHR$(13)+"goto220"+CHR$(13)
200 CLS:PRINT"Deseja alterar a função (S/N) ? ":PRINT:CLEAR:DIM A!(255)
210 A$=INPUT$(1):IF A$="S" THEN LIST 220 ELSE IF A$<>"N" THEN 210
220 DEF FNF(X)=COS(X)
230 '
240 ' DEFINIÇÃO DO INTERVALO
250 '
260 CLS:PRINT"Definição do intervalo:",,,
270 INPUT"Início: ";XI!:INPUT"Final: ";XF!
280 PRINT:PRINT"Intervalo [";XI!;",";XF!;"]",,,
290 PRINT"Confirma (S/N) ? ";
300 A$=INPUT$(1):IF A$="N" THEN 260 ELSE IF A$<>"S" THEN 300
310 '
320 ' CÁLCULO DOS F(X)5
330 '
340 LOCATE 0,17:PRINT"Aguarde ..."
350 S!=(XF!-XI!)/255
360 I!=XI!
370 FOR C%=0 TO 255
380  A!(C%)=FNF(I!)390  I!=I!+S!
400 NEXT C%
410 '
420 ' CÁLCULO DO MÁX. E MÍN. ABSOLUTOS
430 '
440 HY!=1E-32:LY!=1E+32
450 FOR C%=0 TO 255
460  IF A!(C%) > HY! THEN HY!=A!(C%)
470  IF A!(C%) < LY! THEN LY!=A!(C%)
480 NEXT C%
490 '
500 ' CÁLCULO DOS MÁXIMOS E MÍNIMOS
510 '
520 PY=(255*ABS(XI!))/(XF!-XI!)
530 PX=(192*ABS(HY!))/(HY!-LY!)
540 T!=HY!-LY!
550 '
560 ' PLOTAGEM DO GRÁFICO
570 '
580 SCREEN 2
590 LINE (0,PX)-(255,PX)
600 LINE (PY,0)-(PY,191)
610 FOR C%=1 TO 255
620    LINE(C%-1,192*(HY!-A!(C%-1))/T!)-(C%,192*(HY!-A!(C%))/T!)
630 NEXT C%
640 A$=INPUT$(1)
650 IF A$<>" "THEN790
660 OPEN"GRP:"AS#1
670 SPRITE$(1)=CHR$(193)+CHR$(225)+CHR$(113)+CHR$(57)+CHR$(29)+CHR$(15)+CHR$(7)+CHR$(255)
680 X%=0:Y%=0
690 IF X%>255 THEN X%=255
700 IF X%<0 THEN X%=0
710 PUTSPRITE 0,(X%-8,192*(HY!-A!(X%))/T!-8),2,1
720 IF STICK(0)=7 THEN X%=X%-1
730 IF STICK(0)=3 THEN X%=X%+1
740 IF INKEY$=CHR$(13) THEN 780
741 A$=INPUT$(1)
750 LINE(0,0)-(97,9),4,BF
760 PSET(0,1),1:PRINT#1,A!(X%)
770 GOTO 690
780 CLOSE #1
790 SCREEN0:PRINT"Deseja continuar (S/N) ? ";
800 A$=INPUT$(1):IF A$="S" THEN 200 ELSE IF A$<>"N" THEN 800
810 CLS:POKE &HFCAB,0:END
820 RESUME: NEXT
```

CRIPTOGRAFIA

Nos dias de hoje, pode ser de fundamental importância criptografar um programa para proteger-se dos abelhudos. Veja como fazê-lo no MSX.

CARLOS E. A. MOREIRA

Dando prosseguimento ao artigo iniciado pelo nosso amigo J. L. Fonseca, no número anterior desta revista, voltamos a falar sobre criptografia.

O prefixo cripto é derivado do latim "kruptus", o que se traduz como "oculto" Assim sendo, o sentido da palavra criptografia seria "escrita oculta", isto se quisermos levar o sentido ao pé da letra.Como foi citado naquele artigo, a criptografia já era conhecida pelo menos desde a época de Júlio Cesar, porém outros fatores levam a crer que isto é ainda mais antigo e pode vir a datar de 1500 anos A.C.

Grandes desenvolvimentos das diversas técnicas de criptografar são conseguidos, em geral, quando ocorrem os conflitos entre nações, isto devido ao fato de que nestas ocasiões uma grande quantidade de mensagens secretas são intercambiadas. Isto pôde ser notado com certa facilidade quando das deflagrações das duas Grandes Guerras.

Deixando a História de lado, partiremos, agora, para descrevermos os dois métodos mais comumente utilizados na técnica de criptografar.

O primeiro deles consiste em simples substituição de caracteres.Por exemplo : substitui-se a letra "a" pela letra "y", a letra "b" pela letra "q", e assim por diante. Claro que tal substituição deve obedecer a uma regra básica e seguir uma tabela de conversão previamente definida. Para elucidar um pouco mais esta idéia vamos supor uma tabela de conversão e fazer a codificação de uma dada frase. Tome a seguinte tabela de conversão:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
EMPLQGXACIUZYWTRHSBVKNJODF

A frase "CPU A REVISTA DO MSX" ficaria assim codificada: "PRK E SQNCBVE LT YBO". Logicamente, se você conhece a tabela de conversão usada para tal codificação, é evidente a facilidade de compreensão do que está escrito. Porém, para aquele que não tem em mãos tal tabela, verifica-se a dificuldade ou a quase impossibilidade deste entender o que está escrito. Este tipo de codificação foi bastante difundido na época de Jùlio Cesar.

Com a chegada do computador eletrônico,outros métodos foram desenvolvidos. O mais popular deles é o que faz a manipulação de bits. Para quem ficou "boiando", não entre em desespero (pelo menos ainda), pois eu vou explicar como se dá este processo.

Bem, sabemos que cada caracter é representado por um número, isto é, segue um padrão. O padrão seguido pela linha MSX é o ASCII (American Standard Code for Interchange Information).

Assim, por exemplo, o caracter "A" em ASCII é associado ao nùmuro 65, o caracter "B" ao 66 e assim por diante. Quando lemos um caracter, por exemplo de um arquivo, podemos facilmente obter o código ASCII deste. Basta, para isto, usarmos a função certa em cada linguagem de programação. Agora, vocês devem estar se perguntando o que tem isto a ver com manipulação de bits. Respondo que tem muito, pois sabemos que os números, cada um deles, são formados a partir de Bytes e sabemos que um byte é o agrupamento de oito bits. Então, o número 65 decimal é representado, a nível de máquina, em um byte desta forma: "01000001". A manipulação a nível de bit é feita usando a operação lógica XOR. Entenda isto com o OU EXCLUSIVO.

A tabela verdade para esta operacão lógica é a seguinte:

A	B	(A XOR B)
0	0	0
0	1	1
1	0	1
1	1	0

O que torna esta operação extremamente interessante no de criptografia é que, se aplicarmos num mesmo operando duas vezes a mesma operação, o operando volta ao estado anterior, ou seja, a ser novamente ele mesmo. Vamos fazer uma operação do tipo XOR para esclarecermos as dúvidas. Por exemplo:

```
65(d)          -> 01000001(b)
66(d)          -> 01000010(b)
RESULTADO      -> 00000011(b)
```

Fazendo, agora, a operação lógica do resultado da operação novamente com o numero 66(d), retornamos ao nosso valor original 65(d), como podemos ver:

```
RESULTADO      -> 00000011(b)
66(d)          -> 01000010(b)
65(d)          -> 01000001(b)
```

A utilidade de se criptografar programas e/ou arquivos está no fato de que podemos proteger estes de bisbilhoteiros e, desta forma, preservarmos a privacidade de certos programas e/ou arquivos que venham a interessar somente a você mesmo ou a pessoas por você autorizadas.

A proteção de certo software que deve ser utilizado somente por pessoas devidamente credenciadas está incluída no fator acima descrito.

Com esta finalidade, desenvolvi um software que faz exatamente isto, ou seja, lê um arquivo do drive e o criptografa, deixando-o assim totalmente ilegível, protegendo-o desta forma de ser acessado por pessoas que não têm autorização para tal.

A linguagem escolhida foi o "C". Isto se justifica devido ao fato das muitas facilidades oferecidas por esta linguagem e por ser ela a mais adequada a este tipo de aplicação.

O programa listado abaixo foi editado em um editor de textos comum e compilado pelo AZTEC C. Os testes feitos com este foram completamente satisfatórios, criptografando e descriptografando os arquivos normalmente como esperado.

Edite o programa abaixo listado, salve-o sob o nome de 'kriptor.c', compile-o, e se tudo estiver ok, faça os seguintes testes e veja os resultados.

1. Digite simplesmente kriptor e dê <enter>. Ele irá limpar a tela e emitir a mensagem de falta de operando e mostrar a sintaxe certa.

2. Digite um arquivo texto para podermos experimentar o criptografador . Dê o nome a este arquivo de teste.txt. Chame o programa "kriptor" da seguinte forma:

A>kriptor teste.txt teste.kpr, dê <enter>. Ele deverá perguntar a voçê qual a palavra chave, ou seja, a senha para que ele possa usá-la na criptografação. Voçê deverá guardar esta palavra para fazer a posterior descrip-tografação. Use a palavra "teste" de <enter> e a operação será iniciada e rapidamente terminada. Tente dar um 'type' no arquivo "teste Kpr".

Notou o que ocorreu?

```
#include <stdio.h>

main(argc,argv)
int argc;
char *argv[];

{
FILE *read, *write;
        int c, x, i=0, y;
        char d, e, s[];

        if (argc<3) {
        printf("\fFalta Operando");
        printf("\nSintax Correta e:");
        printf("\nKriptor Arq_Fonte
Arq_Destino");
        exit(0);
        }

        if((read=fopen(argv[1],"r"))==NULL)
{
        printf("\fNão Consigo Abrir
Arq_Fonte.");
        exit(0);
        }

        if((write=fopen(argv[2],"w"))==NULL)
{
        printf("\fNão Consigo Abrir
Arq_Destino.");
        exit(0);
        }

        printf("\fEntre Palavra Chave: ");
        gets(s);

        while((c=fgetc(read))!=EOF) {
        if((s[i])=='\0')
  i=0;
 x=(c^(s[i]/4));
 fputc(x,write);
 i++;
 }

        printf("\nOperação Terminada.");
        printf("\nPressione <ENTER>.\n");
        getchar();
        putchar(12);
        fclose(read);
 }      fclose(write);
```

CONSTRUINDO PROGRAMAS

Fazer um programa profissionalmente pode não ser tão difícil quanto você possa imaginar. Neste artigo, o autor mostra alguns passos importantes que devem ser seguidos.

BRUNO MARRUT

Apesar de poder parecer estranho, o fundamental em um programa é que ele funcione, o que pode ser fácil de ser averiguado, dependendo da extensão do mesmo. Todos nós já ouvimos falar, pelo menos uma vez, de erros em grandes programas que vieram a ocasionar grandes prejuízos.

É importante que o programador observe as características sobre as quais o programa deve funcionar, não se deixando levar aos extremos no desenvolvimento de pequenos detalhes, perdendo, com isso, o objetivo principal. Um programa não pode apresentar soluções parciais aos problemas para os quais foi desenvolvido, o que, certamente, irá gerar descontentamento por parte do usuário.

DOCUMENTAÇÃO

Para o programador que está desenvolvendo determinado programa pode parecer desnecessário efetuar comentários do mesmo, parecendo-lhe que procedendo assim estaria perdendo tempo.

Depois de pronto, provavelmente, o programador ainda será capaz de descrever os passos do programa, uma variável, e efetuar alterações que possam vir a ser necessárias. Passado algum tempo, estas informações já não poderão ser fornecidas com tanta certeza, tornando uma tarefa que poderia ser simples em algo trabalhoso.

A falta de comentários em um programa fará com que uma correção ou alteração que tenha que ser feita por uma outra pessoa, seja uma operação até mesmo impossível de ser realizada.

A documentação pode ser feita de duas formas: através de manuais que conterão todas as informações para que o usuário possa utilizar os recursos do programa e que também poderá conter a descrição de algoritmos, fluxogramas, etc; e a documentação que será feita dentro do próprio programa.

Lembre-se que os programas necessitam de facilidade para manutenção e modificação para que possam vir a atender a novas especificações.

OS ERROS

Cabe ao programador a responsabilidade do bom funcionamento ou não do programa.

Hoje em dia não se aceitam mais erros em um programa com base na teoria de que o programador é humano.

Os erros, geralmente, ocorrem por falhas na digitação, não cumprimento das especificações, sintaxe da linguagem, etc.

DESENVOLVENDO UM PROGRAMA

O desenvolvimento de um programa deverá passar pelas seguintes fases:

Análise do problema

Solução

A solução como programa

Testes

Documentação

Na análise do problema, o programador deverá entendê-lo para que possa passar à solução. Um problema mal interpretado irá gerar erros que, depois, serão difíceis de serem encontrados.

A solução irá depender da criatividade do programador, existindo métodos que poderão auxiliá-lo, como, por exemplo, o método "Top-down".

É comum observarmos programadores que não fazem um profundo estudo do problema e uma criteriosa avaliação da solução obtida, partindo logo para a montagem do programa, gastando tempo, depois, na procura dos erros.

Entendido o problema e achada a solução, devemos transformar a solução em um programa, o que poderá ser extremamente fácil, sendo esta etapa quase que mecânica para o bom programador que conhece bem a linguagem de programação.

Os testes efetuados em um programa irão mostrar a presença de erros e não a sua ausência. Portanto, não se desanime nesta etapa final do processo de criação de um programa.

A única forma de testarmos um programa consiste em testar todos os casos possíveis, ou seja, todas as combinações possíveis, sendo que esta tarefa pode consumir um bom período de tempo, dependendo do programa, cabendo ao programador eliminar situações de teste exdrúxulas.

O teste de um programa deve ser feito a partir das especificações e não do próprio programa. A elaboração do manual seria a última etapa do desenvolvimento, onde devem estar contidas todas as informações necessárias ao bom aproveitamento do programa.

A ESCOLHA DE UMA LINGUAGEM

Para que as pessoas se comuniquem com os computadores é necessário uma linguagem de programação que irá informar ao computador exatamente o que queremos que ele faça, em uma linguagem que ele possa entender.

Existem enúmeras linguagens de programação, sendo que entre as mais comuns podemos citar o Basic, Cobol, Fortran e Assembly, sendo esta última também chamada de Linguagem de máquina ou Assembler.

Efetuar a escolha da linguagem certa também é um fator importante, pois determinados sistemas poderão ser feitos com maior facilidade e apresentar uma maior rapidez em certa linguagem.

Para a linha MSX já estão disponíveis a maioria das linguagens em disco, sendo que os usuários de fita ainda contam com um nùmero pequeno de possibilidades de escolha.

Basicamente, podemos dividir as linguagens existentes em três grandes grupos que são as de extra-alto nível, alto nível e as de baixo nível.

As linguagens de baixo nível, que, como exemplo, poderíamos citar o Assembler, estão voltadas mais para o próprio computador do que para o programador. As linguagens de alto nível, como o Basic, estão voltadas para o usuário.

Exemplificando: a diferença entre uma linguagem de alto nível e uma de baixo nível é a diferença entre dar a um mecânico um esquema detalhado da máquina que ele deve montar (alto nível) ou ir lhe dizendo onde, exatamente, cada peça deve ser colocada (assembler).

Logicamente, cada um destes grupos possui vantagens em relação ao outro.

O Assembler apresenta a característica de ser um programa mais compacto e de ter maior velocidade.

É dado o nome de linguagem de nível extra-alto a programas sofisticados que são autoprogramáveis, a exemplo do Lotus 1-2-3, do dBase II, etc. Você poderá fazer um programa de mala direta utilizando o Basic ou através do dBase.

A vantagem da utilização de linguagens de nível extra-alto é a facilidade de se escrever os programas e testá-los rapidamente, não sendo necessário que o programador fique preso a detalhes, como o gerenciamento de arquivos em disco.

As linguagens de alto nível são as mais utilizadas hoje em dia.

O Cobol e o Fortran foram as primeiras linguagens de programação desenvolvidas, sendo o Cobol orientado para aplicações comerciais e o Fortran indicado para cálculos. Desde o surgimento destas duas linguagens, muita coisa nova foi introduzida.

O Basic é a linguagem mais utilizada em microcomputadores, sendo uma linguagem de fácil aprendizado. Pode ser usada tanto em computadores de grande porte como em microcomputadores e, principalmente, é uma linguagem padronizada que permite ao programador desenvolver e alterar programas sem muito esforço.

Outras linguagens como Pascal, C, Forth também possuem suas características especiais.

Os programas elaborados em uma linguagem de alto nível que possui instruções mais semelhantes com a linguagem humana devem ser traduzidos para a linguagem de máquina antes de serem executados. Cada comando da linguagem é traduzido em muitas instruções de linguagem de máquina, ou seja, em informações bem mais detalhadas.

A este processo denomina-se compilação ou interpretação.

O compilador efetua a sua tradução antecipadamente, antes que o programa traduzido seja executado, ao passo que o interpretador vai fazendo sua tradução à medida que o programa vai sendo executado.

A compilação, a princípio, apresenta vantagens sobre a interpretação, como maior velocidade de execução e pelo fato do programa compilado ser uma versão otimizada. A interpretação também possui suas vantagens , sendo a principal delas a facilidade para alterar o programa durante sua execução.

PASCAL PARTE 1

ANTONIO F. S. SHALDERS

Iniciaremos, este mês, um curso sobre a linguagem Pascal, uma das mais versáteis e simples linguagens de programação existentes para microcomputadores.

Este curso exige que o leitor tenha algumas noções de computação (não especificamente BASIC), pois começar a ensinar computação da estaca zero é uma tarefa muito demorada e exigiria artigos por demais extensos.

O que proponho é transmitir ao leitor uma nova linguagem, pois muitos problemas computacionais não são satisfatoriamente solucionados com o uso do BASIC.Seria bom se você conhecesse os aspectos fundamentais de computação, como variáveis e loops.

O Pascal foi criado por Niklaus Wirth para ser uma poderosíssima ferramenta de ensino na área de computação.

Um ponto forte do Pascal é que esta linguagem é estruturada e é possível a criação de programas recursivos. É uma linguagem muito mais poderosa que o BASIC e, embora não pareça `a primeira vista, é muito simples.

Para o programador acostumado apenas com o BASIC, o aprendizado do Pascal será um pouco mais difícil, pois o raciocínio usado para a elaboração de programas em Pascal é dife
rente do usado em BASIC.

O Pascal possui recursos que o BASIC nem sonha, como por exemplo o uso de funções e procedimentos definidos pelo usuário. Isto significa que podemos, literalmente, criar uma nova palavra que passará a chamar uma determinada subro tina ou uma função matemática complexa, por exemplo.Os programas em Pascal são muito mais rápidos que os escritos em BASIC, além de terem aparência mais profissional.

O nosso curso destinar-se-á à versão Turbo Pascal 3.0 desta linguagem. Convém adquirir um guia de operação em forma de folheto e algum manual para este compilador.

Existem várias publicações a respeito deste compilador, sendo a da McGraw-Hill uma das melhores que conheço (Turbo-Pascal guia do usuário) .

Aos possuidores de outras versões do Pascal, como o Pascal ISO e o Hisoft Pascal, informo que, eventualmente, serão necessárias modificações nos programas apresentados, mas isso não impede que sigam o curso, pois serão apresentadas todas as características comuns a estes compiladores.

É bom lembrar que este não é um curso de operação do Turbo Pascal. Nós, apenas, o tomaremos como referência, sendo de responsabilidade do leitor a sua aquisição, assim como dos respectivos manuais e/ou referências.

Se você possui disk-drive, vale a pena mencionar que o Turbo Pascal é composto de vários subprogramas, cada um com uma função específica:

TURBO.COM :
É o compilador em si.

TURBO.MSG :
Contém as mensagens deerro.

TURBO.OVR :
Permite a execução deum programa a partir do Turbo.

TLIST.COM :
Gera listagens .(Opcional).

TINST.COM :
Instalador de tela e teclado. (Opcional)

GRAPH.P :
Rotinas gráficas(Opcional).

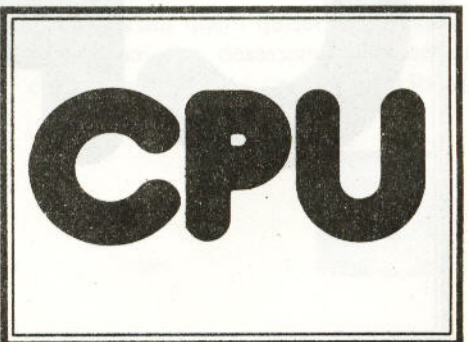
A versão mais recente para micros de oito bits do Turbo Pascal é a 3.0, sendo que versões superiores a esta são alterações da mesma feitas pelo instalador.

Uma grande vantagem do Turbo é que é possível a criação de um programa do tipo executável direto em linguagem de máquina (.COM).

É uma linguagem que vale a pena ser aprendida, pois nem sempre o desempenho de um programa em BASIC é satisfatório, obrigando-nos a convertê-lo para uma linguagem mais rápida.

O Pascal pode ser considerado uma linguagem boa para quase todas as aplicações, porém desde que a velocidade exigida não seja muito alta. Neste caso, devemos apelar para Forth ou Assembly, e se o caso for muita dependência com o hardware da máquina, podemos recorrer à linguagem C .

É uma lingagem relativamente boa para aplicações gráficas e matemáticas e outras aplicações profissionais.

Feita esta abordagem inicial, convém estudarmos um pouco sobre a estrutura de um programa em Pascal.

A ESTRUTURA DO PASCAL

É drasticamente diferente da de um programa em BASIC, pois em Pascal devemos dividir o programa em blocos estruturais, o que não é usual em BASIC.

A primeira parte do programa chama-se cabeçalho, e deve conter o nome do programa.

Ex: PROGRAM TESTE ;

Esta linha deve ser a primeira do programa. Note que termina em um ponto e vírgula, este necessário sempre (ou quase) ao final de uma linha.

Em segundo lugar, vem a área de definições de tipos e variáveis. Esta área serve para definirmos como uma determinada variável deverá ser tratada.

Os tipos de variáveis são: inteiras (INTEGER), reais (REAL), alfanuméricas(CHAR), lógicas (BOOLEAN) ou byte (BYTE), cada uma com um uso específico.

Um exemplo de área de declarações de variáveis é dado abaixo:

```
VAR A,B : REAL ;
```

Isto significa que as variáveis A e B deverão ser tratadas como reais. Declarações semelhantes deverão ser feitas para outros tipos.

Analisaremos, agora, os tipos de variávejs e suas características:

VARIÁVEIS INTEIRAS

A sua principal característica é que estão obrigatoriamente contidas no intervalo que vai de -32768 a 32767, o que nos dá 65536 números diferentes. São usadas, principalmente, no controle de loops do tipo FOR (isto será explicado em outra lição).

São possíveis as operações de soma (+), subtração (-), multiplicação (*), divisão inteira (DIV) e resto (MOD).

Estas operações serão explicadas mais adiante.

VARIÁVEIS REAIS

São usadas apenas em cálculos matemáticos ou quando o valor desejado está fora da faixa dos inteiros.

Os reais são mostrados em notação científica.

As operações possíveis com os reais são: soma, subtração e multiplicação (como no caso dos inteiros) e a divisão real (/).

A seguir, são dados alguns exemplos:

```
15 / 2      = 7.5 (REAL)
15 DIV 2    = 7 (INTEGER)
15 MOD 2    = 1 (INTEGER)
```

VARIÁVEIS ALFANUMÉRICAS

Este tipo é análogo ao do BASIC, armazenando um caracter qualquer. Uma string em Pascal é definida como um vetor de caracteres; logo, quando uma string for definida, o seu comprimento deverá ser especificado.

As operações possíveis com strings são muitas e serão demonstradas oportunamente.

É mostrado, abaixo, um exemplo de definição de uma variável como uma string de 100 caracteres :

```
VAR A : STRING [ 100 ] ;
```

VARIÁVEIS BOOLEANAS

São consideradas booleanas ou lógicas as variáveis que podem conter apenas dois tipos de informação: verdadeiro (TRUE) ou falso (FALSE). Seu uso é principalmente para testes.

VARIÁVEIS BYTE

Esta categoria é um tipo especial, que pode conter um valor inteiro qualquer entre 0 e 255.

Seu uso é feito apenas em condições especiais que serão oportunamente abordadas.

Na próxima lição, serão apresentadas as áreas de definições de procedimentos e funções e a área do programa principal, além das primeiras noções sobre loops com alguns exemplos práticos. Até lá !

MATEMÁGICA
LÓGICA DE VALORES

J.L. FONSECA

Na coluna de hoje, vamos falar de um assunto que, embora aparentemente seja muito teórico, pode ter aplicações práticas extremamente interessantes.O assunto de hoje é sobre a lógica de valores múltiplos.

Todos os que mexem com computadores estão acostumados a usar expressões lógicas e sabem, certamente, que só existem dois valores lógicos que são o 0 e o 1, que geralmente, representam as expressões falso e verdadeiro. Para a maioria dos problemas de computação isto é verdade, mas, quando começamos a mexer com o mundo real, reparamos que nem sempre as coisas são tão simples.

Se queremos que o nosso programa leve em conta situações reais, deparar-nos-emos, frequentemente, com situações em que uma pergunta não tem uma resposta taxativa, mas sim uma resposta aproximada ou subjetiva. Exemplificando: uma pessoa pode ser mais ou menos alta, o café pode estar mais ou menos quente, um filme ser mais ou menos bom, etc.

Quando um programa nos faz perguntas do tipo acima citado, não podemos responder com um simples sim ou não e esperar resultados significativos, pois a nossa resposta tão pouco o é.Por exemplo: se tivermos um programa que acesse um banco de dados com as características de todos os livros da nossa biblioteca e lhe pedirmos que nos ache um artigo sobre eletrônica que seja curto,ele pode selecionar artigos de duas ou de trinta páginas, dependendo do conceito de curto de quem fez o programa, ou, no caso dele nos perguntar quantas páginas tem um artigo curto, só nos mostrar os artigos com aquele número de páginas ou um número inferior e deixar de fora um artigo interessante só porque ele tem uma ou duas páginas a mais.

Se estivesse usando uma lógica de valores múltiplos, no entanto, o programa nos daria uma lista de artigos ordenada de acordo com a probabilidade de atenderem ao nosso critério, podendo, inclusive, nos dar uma estimativa de quanto do nosso critério cada artigo satisfaz.

O exemplo acima foi para uma única condição mas poderia ser para qualquer número delas, ligadas pelos conectivos lógicos normais, pois ainda assim ele nos daria o grau de confiança depositado em cada sugestão.

A frase grau de confiança define com perfeição as respostas dadas pelas operações lógicas, ou seja, nos dá uma estimativa de quão confiável uma afirmação é. No caso da lógica comum nós temos sempre afirmações totalmente verdadeiras ou totalmente falsas, mas na lógica de múltiplos valores, como na vida real, as verdades podem ser mais ou menos verdadeiras.

Na lógica de múltiplos valores, uma afirmação pode ter um valor que varia entre 0 e 1, sendo o seu valor um nùmero real.Por exemplo: a frase "o dia está lindo" pode ter um valor de 0,5 caso o dia esteja regular, ou 1 caso seja o dia mais lindo da sua vida, ou 0 no caso de ser um dia horrível.

À primeira vista, as operações lógicas comuns não são aplicáveis à lógica de valores múltiplos, mas, se nós definirmos as operações lógicas de um modo mais amplo, veremos que os novos operadores podem trabalhar com lógica comum do mesmo modo que com valores mùltiplos e imprecisos.

Vamos, pois, definir estas operações de modo análogo às operações da lógica comum. Comecemos pela função AND ou "E", que dá como resultado o valor mínimo dos dois valores de entrada,ou seja, 0 E 1 dá 0, o que está de acordo com a lógica comum e com o bom senso, pois uma conjunção de duas afirmações é tão verdadeira quanto a menos verdadeira das duas. Por exemplo: a frase " as laranjas são redondas e são doces", com uma probabilidade de 0,9 de as laranjas serem redondas (caso bastante comum) e de 0,5 delas serem doces (o que já não é tão comum) nos dá uma probabilidade ou confiabilidade de 0,5 de que uma laranja qualquer seja doce e redonda simultaneamente.

A função OR ou "OU" é definida como sendo igual ao máximo valor de entrada, o que, uma vez mais, está de acordo com a lógica comum e com o bom senso, pois se dois acontecimentos distintos podem acontecer, a probabilidade de que pelo menos um deles aconteça é igual à probabilidade do mais provável. Como exemplo temos a frase "posso acertar na loto ou ir trabalhar na segunda-feira" com a probabilidade de 0,001 de acertar na loto e a probabilidade de 0,99 de ir trabalhar na segunda-feira. Temos como resultado uma confiabilidade de 0,99 de que, pelo menos, iremos trabalhar na segunda-feira.

A última função básica é a função NOT ou "NÃO", que tem como valor de saída 1 menos o valor de entrada, ou seja, a probabilidade de algo não acontecer é o complemento da probabilidade desse algo acontecer. Exemplificando: se temos a frase "todos os gatos são felinos" e temos uma confiabilidade de 0,8 na nossa afirmação, a nossa confiança em que nem todos os gatos sejam felinos é de 1-0,8, o que é igual a 0,2.

As três funções anteriores são funções básicas que se constituem nos blocos construtivos para funções mais complexas como a função XOR ou "OU EXCLUSIVO", que é uma função que escolhe entre duas opções mutuamente exclusivas e que pode ser expressa como (("a" E "b") OU ((NAO "a") E (NAO "b"))).

Temos falado, nos parágrafos anteriores, em probabilidades, mas devemos fazer uma pequena correção para que não haja confusão.

As probabilidades citadas não são, na verdade, probabilidades no sentido comum da palavra, mas sim graus de confiança nas afirmações, o que dá resultados mais conservadores do que as probabilidades estudadas em estatística (bayesianas), mas, por isso mesmo, mais úteis quando usadas em sistemas de aconselhamento automatizado, onde é preferível pecar pelo excesso de cuidado do que pela falta do mesmo.

Como foi dito no início, não temos na coluna deste mês programas, mas temos a fundação para criar, no futuro, programas inteligentes sem usar técnicas extremamente avançadas, e, ainda assim, termos resultados bastante compensadores.

Vamos, pois, terminar por aqui e até o próximo número.

MSX WORD 3.0

Análise do novo software da Cibertron

O MSX WORD é um dos editores de texto mais utilizados pelos usuários do MSX devido às facilidades que apresenta de operação.

Algumas características, que devem ser levadas em conta na hora da escolha de um processador de textos, são as seguintes:

Acentuação em português. Os acentos são obtidos do mesmo modo que em uma máquina de escrever.

Controles de modo de impressão.

Configuração para as impressoras Gafix MTA e 80/100, sendo que o Set de caracteres é totalmente configurável, possibilitando o uso de Back Space para as impressoras que não possuem os acentos.

Os arquivos, que tanto podem ser gravados em fita ou disco, são gravados em ASCII.

Compatível com até dois drives de 5 1/4" ou 3 1/2". Pode ser utilizado como editor de programas fonte de outras linguagens, visando à futura compilação.

Possui conversor de 80 colunas para 64 (formato utilizado pelo MSX WORD), visando adaptar textos de outros editores, inclusive os do PC para edição no MSX WORD.

Quem possui uma versão do MSX WORD anterioir à versão 3.0 certamente já se deparou com algumas dificuldades que nesta nova versão foram eliminadas, tornando o editor mais fácil de se utilizar.

Das alterações introduzidas podemos destacar a rotina de carregamento que foi alterada e que não mais obriga o usuário a manter a tecla CTRL pressionada, a fim de liberar mais memória e, com isso, limitando a apenas um drive a disponibilidade de gravação. Nesta nova versão, o usuário pode selecionar em qual dos drives deseja que o arquivo seja lido ou gravado, caso a configuração do seu sistema possua dois drives.

As versões anteriores do MSX WORD gravam o arquivo texto em ASCII, contudo não era possível utilizar o arquivo em outros editores de texto, que também gravam seus arquivos em ASCII. Através de alterações efetuadas na rotina de gravação, o que também reduziu o tempo de gravação em até 30%, a nova versão efetua a gravação do arquivo texto em ASCII padrão, vamos dizer assim, permitindo que o arquivo possa ser editado em qualquer editor que utilize este padrão de gravação.

A rotina de detecção de erros foi melhorada e não é mais permitido que um arquivo seja gravado em disco caso já exista um arquivo com o mesmo nome no diretório, eliminando, assim, a possibilidade de se perder um arquivo acidentalmente, gravando-se um outro no lugar.

Um dos pontos fracos do MSX WORD era, sem dùvida alguma, sua rotina de impressão, que era totalmente controlada pelo usuário, não permitindo que fossem efetuadas várias cópias de uma só vez, sendo que cada cópia devia ser solicitada em separado, tornando o trabalho mais demorado e cansativo. Na nova versão este problema também foi resolvido.

Devido às facilidades que oferece, bem como a apresentação do texto na tela, o MSX WORD é um editor de textos que, certamente, não pode faltar na sua biblioteca de software, ainda mais se você tem que datilografar textos com frequência.

Apresentação:

O programa nos foi enviado pela Cibertron em disco de 5 1/4", devidamente embalado e protegido, sendo que podemos notar que é dispensado por parte do fabricante um cuidado todo especial para que o software chegue em perfeitas condições ao usuário.

Acompanhando o disquete, recebemos o manual de instruções, bem elaborado, totalmente detalhado e com cópias das telas do programa, onde são analisados todos os comandos do programa, sua utilização com impressoras, gravação e leitura de arquivos, acentuação, utilização com compiladores, etc.

O manual possui índice que facilita ao usuário a consulta, bem como análise de todos os possíveis erros que podem ocorrer na sua operação.

É também no manual que o fabricante dá a garantia do produto, por um prazo de 90 dias.

Acompanha o programa um outro manual que vem a ser uma "Iniciação rápida com a MTA", onde são fornecidos exemplos de utilização do software com este tipo de software.

Brevemente, a Ciência Moderna lançará um manual para todas as versões do MSX, inclusive a 3.0.

LIVROS

LINGUAGEM DE MÁQUINA
Editora Aleph
Rossini - Figueiredo
160 páginas

A aprendizagem de uma nova linguagem é uma tarefa nem sempre fácil, principalmente em se tratando de Linguagem de Máquina.

O livro da Editora Aleph exige, por parte do leitor, um certo esforço, mesmo para aqueles programadores que já possuem alguma experiência em Basic, pois uma nova linguagem será ensinada e um novo vocabulário será adquirido. Novos sistemas de numeração também serão utilizados.

O livro foi dividido em nove capítulos, onde são abordados os seguintes assuntos:

Conhecendo melhor o Z-80, Primeiras instruções, Instruções aritméticas, Deslocamento de blocos, Saltos e sub-rotinas, Instruções Lógicas e Operações com Bits, Finalizando as instruções, Instruções secretas do Z-80 e Primeiras Aplicações.

Os apêndices são 2 e referem-se à conversão de sistemas de numeração e tabela de instruções do Z-80.

Um dos capítulos que chama a atenção é o que se refere às instruções secretas do Z-80, que vêm a ser os "buracos" que aparecem numa tabela de mnemônicos após CB e após ED.

Estas instruções não constam nos manuais, não fazendo parte das tabelas dos programas Assembler e Disasembler, podendo, contudo, ser utilizadas.

A apresentação do livro segue o padrão da Editora Aleph, com várias ilustrações, tabelas e exemplos que facilitam a assimilação dos conceitos por parte do leitor, tornando o livro de leitura agradável.

ROTINAS FINANCEIRAS
Ciência Moderna
Armando Oscar Cavanha Filho
166 páginas

O MSX vem ganhando cada vez mais usuários que descobrem as potencialidades deste micro e verificam que ele pode ter inúmeras aplicações na área profisional.

A proposta do livro "Rotinas Financeiras" é a de abordar situações comuns da matemática financeira e mostrar como, através de conceitos e técnicas de programação, elas podem ser utilizadas com o auxílio de um micro computador.

Os programas apresentados estão escritos no MSX BASIC, sendo que nos apêndices os mesmos poderão ser encontrados em sua versão para o BASIC do IBM-PC.

Hoje em dia, com a inflação que se apresenta, saber aplicar o dinheiro ou saber se determinado negócio é vantajoso ou não, é tão importante quanto realizar o negócio propriamente.

Neste livro temos os seguintes capítulos: Pagamento Único; Série Uniforme de Pagamentos; Juros; Série não Uniforme; Depreciação; Ações na Bolsa de Valores; Controle de Clientes na Empresa; Alugar ou Comprar; Tabela de Juros; Análise de Risco; Rotina Gráfica de Fluxo de Caixa; Curvas de Juros; e Calendário Financeiro.

No início de cada capítulo, a autor descreve o assunto proposto, fornecendo os conceitos, fórmulas, gráficos e todas as demais informações necessárias para uma boa assimilação por parte do leitor. Os programas são apresentados como uma decorrência da situação que está sendo analisada, sendo, portanto, fácil o entendimento de sua estrutura.

O MEU PRIMEIRO LIVRO MSX
Editora McGraw-Hill
Tony Marriot
131 páginas

Como sugere o título, este lançamento da McGraw Hill é dirigido aos iniciantes, que querem entender o funcionamento de um microcomputador, explicando de maneira simples como conectar, testar e utilizar o seu MSX.

No primeiro capítulo do livro, é feita uma análise dos termos mais comumente utilizados em computação, tais como: Software e Hardware.

No segundo capítulo, que tem o título de "Juntando Tudo", são descritas as conexões do MSX, como cassete, display de vídeo, etc.

O terceiro capítulo diz respeito ao teclado e é explicado o funcionamento das diversas teclas, como as de função, HOME/CLS, etc.

A partir do quinto capítulo, o autor dá início à explicação dos comandos de programação e sua utilização, iniciando com o comando PRINT e apresentando as mensagens de erro.

No capítulo 8 são mostrados os primeiros programas, onde os comandos vão sendo apresentados com exemplos de programas.

Os capítulos 9 e 10 são reservados somente para gráficos, onde também temos exemplos de programas ilustrando os comandos.

A música no MSX é abordada no ùltimo capítulo do livro.

A utilização, sempre que necessário, de programas utilizando os comandos explicados ,facilita em muito a aprendizagem por parte do leitor, que terá, neste livro, um grande auxílio para a sua iniciação no mundo da computação

SOS FELINO

SILVIO CHAN

AJUDE O BOMBEIRO A SALVAR OS GATINHOS QUE ESTÃO PRESOS NA ÁRVORE.

Neste jogo você é um bombeiro e deverá resgatar os gatos presos em cima de uma árvore.

Para atrapalhá-lo, existem insetos espalhados pela copa da árvore que, ao serem tocados, fazem com que você perca uma vida (life).

O tempo também poderá atrapalhá-lo. A cada intervalo de aproximadamente 11 segundos (1 unidade de tempo), sua vida diminuirá em um ponto.

Cada gato, ao ser resgatado, somará um ponto à sua vida.O aparecimento dos gatos e insetos é aleatório.

Para se movimentar, use as setas laterais. Para subir, movimente o bombeiro até a extrema esquerda. Ele, automaticamente, subirá um galho. Quando você estiver no ùltimo galho, poderá descer levando o bombeiro até a direita.

Caso sobreviva durante dez unidades de tempo, passará para o próximo nível.

Neste jogo, o fator sorte irá influir expressivamente. Cada partida será diferente da anterior, visto que seu desempenho dependerá do nùmero de gatos e insetos que o computador apresentar durante um nível.

```
20 'SOS FELINO — (c) Esfera
40 'Copyright by SChan. 1988
70 '
80 'I N I C I A L I Z A R
90 '
100
CLEAR500:COLOR15,4,4:KEYOFF:SCREEN1,2,0:WIDTH32:
C=RND(-TIME)
110 '
120 ' DEFINIR BLOCOS GRAFICOS
130 '
140    FORI=384TO983:A=VPEEK(I):B=AORA/
2:VPOKEI,B:NEXT
150 DATA170,85,170,85,170,85,170,85
160 DATA255,239,219,189,189,255,103,2
170 DATA36,90,189,24,255,24,126,153
180 DATA66,189,129,165,129,153,129,126
190
FORI=1536TO1543:READA:VPOKEI,A:NEXT:FORI=
1600TO1607:READA:VPOKEI,A:NEXT:FORI=
1664TO1679:READA:VPOKEI,A:NEXT
200 VPOKE8216,97:VPOKE8217,196:VPOKE8219,160
210 '
220 'B O M B E I R O
230 '
240 DATA0000000110000000
250 DATA0000001001000000
260 DATA0000111111110000
270 DATA0000001001000000
280 DATA0000001001000000
290 DATA0000000110000000
300 DATA0000111001110000
310 DATA0001010000101000
320 DATA0001010000101000
330 DATA0001010000101000
340 DATA0001011111101000
350 DATA0001010000100000
360 DATA0000011111100000
370 DATA0000011001100000
380 DATA0000011001100000
390 DATA0001111001111000
400
FORI=0TO15:READA$:B$=B$+CHR$(VAL("&B"+LEFT$(A$,8))):C$=
C$+CHR$(VAL("&B"+RIGHT$(A$,8))):NEXT:SPRITE$(0)=B$+C$
410 '
420 ' DEFINIR VALORES INICIAIS
430 '
440
A$="o4cecdd":B$="o5cecdd":S=10:T=0:N=1:E=.75:F=1:L(1)=
1:L(2)=6:L(3)=11:L(4)=16:L(5)=16:ONINTERVAL=900GOSUB730
450 '
460 'O C E N A R I O
470 '
480
PLAYA$+B$:PLAYA$,B$:IFF>15THENF=15:E=10
490
CLS:X=32:Y=159:LOCATE12,10:PRINT"NIVEL";N:FORI=0TO2000:
NEXT:
FORI=0TO15STEP5:LOCATE2,
I:PRINTCHR$(200);SPC(26);CHR$(200):PRINT;"
";STRING$(3,200);
SPC(24);STRING$(3,200)
500  PRINT"  ";STRING$(2,200);STRING$(12,192);"
";STRING$(12,192);STRING$(2,200):PRINT"
";STRING$(13,200);"
";STRING$(13,200):PRINTTAB(13);STRING$(2,200);"
";
STRING$(2,200):NEXT
510 FORI=640TO767:VPOKEBASE(5)+I,219:NEXT
520
FORI=2TO18:LOCATE15,I:PRINTSTRING$(2,192):
NEXT:LOCATE13,19:PRINTSTRING$(6,192)
530 '
540 'P R O C E S S A M E N T O
550 '
560 INTERVALON:GOTO680
570                  A=STICK(0):B=(Y+1)*4+X/
8:IFA=3ANDX<224ANDVPEEK(BASE(5)+B+2)<>192AND
VPEEK(BASE(5)+B+34)<>192THENX=X+8:GOTO590
580   IFA=7ANDX>16ANDVPEEK(BASE(5)+B-
1)<>192ANDVPEEK(BASE(5)+B+31)<>192THENX=X-
8
590 IFX<=16ANDY>-1THENX=32:Y=Y-40
600 IFX>=224ANDY<159THENX=208:Y=Y+40
610
INTERVALSTOP:C=RND(1)*15:IFC<.75THENLOC
ATEINT(RND(1)*11+4),L(INT(RND(1)*4+1)):PRINTCHR$(209);
ELSEIFC<1THENLOCATEINT(RND(1)*11+
17),L(INT(RND(1)*4+1)):PRINTCHR$(209);
620
C=RND(1)*15:IFC<ETHENLOCATEINT(RND(1)*11+4),L(INT(RND
(1)*4+1)):PRINTCHR$(208);ELSEIFC<FTHENLOCATEINT(
RND(1)*11+17),L(INT(RND(1)*4+1)):PRINTCHR$(208);
630 INTERVALON
640 IFVPEEK(BASE(5)+B+33)=209THENVPO
KEBASE(5)+B+33,32:BEEP:S=S+1:GOTO680
650 IFVPEEK(BASE(5)+B+32)=209THENVP
OKEBASE(5)+B+32,32:BEEP:S=S+1:GOTO680
660 IFVPEEK(BASE(5)+B+33)=208AND
S>0THENVPOKEBASE(5)+B+33,32:S=S-
1:PLAY"c12"ELSEIFVPEEK(BASE(5
)+B+33)=208ANDS=0THEN690:GOTO680
670 IFVPEEK(BASE(5)+B+32)=208AN
DS>0THENVPOKEBASE(5)+B+32,32:S=S-
1:PLAY"c12"ELSEIFVPEEK(BA
SE(5)+B+32)=208ANDS=0THEN690
680 PUTSPRITE0,(X,Y),1,0:GOSUB780:GOTO570
690  INTERVALOFF:LOCATE11,10:PRINT"Fim  do
Jogo"
700 PLAYA$+B$:PLAYA$,B$:FORI=0
TO2000:NEXT:LOCATE9,10:PRINT"Novamente(S/N)"
710 IFINKEY$<>""THEN710
720
A$=INKEY$:IFA$="S"ORA$="s"THENPUTSPRITE0,
(32,32):GOTO440ELSEIFA$="N"ORA$="n"THENSCREEN
0:ENDELSE720
730 INTERVALSTOP:S=S-1:PLAY"a12"
740
IFS<0THEN690ELSET=T+1:GOSUB780:INTERVALON:
RETURN
750 '
760 'L I F E & T I M E
770 '
780  INTERVALSTOP:LOCATE0,22:PRINT"Life";S;"
Time";T;" Nivel";N
790 '
800 ' PASSAGEM DE NIVEIS
810 '
820               IFT=10THENPUTSPRITE0,(-32,-
32):T=0:N=N+1:E=E+.375:F=F+.5:FORI=0TO1000:NEXT:
GOTO480
830 INTERVALON:RETURN
840 ' A$,B$ - MUSICA
850 ' S - LIFE   T - TIME
860 ' E,F - NUMERO DE INSETOS
870 ' N - NIVEL DE JOGO
880 ' L(1-5) - VERTICAL
890 ' A - VALOR DAS SETAS
900  '  B  -  VALOR  NA  VRAM  DA  POSICAO  DO
BOMBEIRO
```

JOGO DA MEMÓRIA

GUILHERME A. L. DA SILVA

Este programa é uma versão do jogo da memória para o MSX. As regras deste jogo já são conhecidas mas, vale a pena ver as adições feitas nesta versão.

Existem duas opções de jogo:

Jogador contra computador:

Nesta opção você terá 15 níveis de dificuldade, sendo o 1 fácil e o 15 o mais difícil. Nos ùltimos níveis, o computador acerta mais e você não vê as figuras. Nos primeiros níveis, a situação é inversa.

Jogador contra jogador:

Nesta opção você joga contra um amigo seu alternadamente e as figuras aparecem no começo.O vencedor será aquele que tiver obtido um maior nùmero de pontos, sendo que cada par vale um ponto, existindo 8 pares de figuras.

As jogadas são feitas a partir do nùmero e da letra que se encontra a figura. Por exemplo: 1F ou 2EHá possibilidade de empate.

O programa

Ao iniciarmos, o programa pede a quantidade de jogadores e, depois, os seus respectivos nomes. Caso seja apenas um jogador, será solicitado também o nível de dificuldade.

Enquanto o programa efetua os devidos cálculos, será desenhado na tela um quadriculado e tocada uma mùsica. Logo após, serão mostradas as figuras, isto dependendo do nível selecionado.

Para aumentar o nível de dificuldade além do normal, de 1 a 15, entre com valores maiores que 15 e delete a linha 1330. O programa irá demorar um pouco mais para efetuar os cálculos.

Podemos, também, aumentar o nùmero de pares, modificando para isto o sistema de coordenadas ON-GOTO.

As principais variáveis do sistema são:

F$ = Matriz das figuras
A = Nùmero da primeira figura a ser sorteada
B = Nùmero da segunda figura a ser sortead
B$()= Matriz das figuras já sorteadas
A$()= Matriz das figuras já sorteadas
N$()= Nomes dos jogadores
JO = Indicador do jogador
J = Nùmero de jogadores
BN = Nível de dificuldade
P() = Placaresw
H$ = Tentativa do jogador
W$ = Primeiro caracter de H$
Q$ = Segundo caracter de H$
TI = Indicador da rotina 1 e da figura
TP = Indicador da rotina 2 e da figura
C$()= Figura escolhida pelo jogador
X = Coordenada da figura na tela
Y = Coordenada da figura na tela
RI = Primeiro nùmero do computador
RP = Segundo nùmero do computador

```
10 REMJogo da Memoria
20 REMGuilherme A. L. da Silva
30 REM22/06/88
40 REMGUARARAPES - S.P.
50 REMPara a linha MSX
60 CLEAR1000:JO=1
70 SCREEN1,,0:KEY OFF:COLOR 15,12,10
80 STOP ON
90 ON STOP GOSUB 1400
100 REM JOGO DA MEMORIA
110 GOTO 390
120 DIM C$(16),A$(16),B$(16),F$(8)
130 F$(1)=CHR$(1)+CHR$(76):F$(2)=CHR$(1)
+CHR$(66):F$(3)=CHR$(1)+CHR$(69)
:F$(4)=CHR$(1)+CHR$(70)
140 F$(5)=CHR$(1)+CHR$(67):F$(6)=
CHR$(1)+CHR$(68):F$(7)=CHR$(1)+CHR$(79)
:F$(8)=CHR$(1)+CHR$(75)
150 SCREEN1
170 PRINTTAB(5)" Jogo da memória "
190 LOCATE 0,5
200 PRINTTAB(8)"!-!-!-!-!"
210 FORO=1TO3
220 PRINTTAB(8)"! ! ! ! !"
230 PRINTTAB(8)"!-!-!-!-!"
240 NEXT
250 PRINTTAB(8)"! ! ! ! !"
260 PRINTTAB(8)"+-+-+-+-+"
270 LOCATE0,15:PRINT"Qualquer tecla para
começar."
280 A$=INKEY$:IF A$="" THEN GOTO 280
290 LOCATE 0,18:PRINTTAB(5);"Estou
calculando":GOSUB 1240
300 FOR I=1 TO 8
310 A=INT(16*RND(-TIME)+1)
320 IF A<>B THEN 330 ELSE 310
330 IF B$(A)<>"" THEN 310 ELSE B$(A)=F$(I
340 B=INT(16*RND(10)+1)
350 IF B<>A THEN 360 ELSE 340
360 IF B$(B)<>"" THEN 340 ELSE B$(B)=F$(I)
370 NEXT
380 Z=0:FORI=1 TO
16:A$(Z)=B$(I):Z=Z+1:NEXT:RETURN
390 REM
400 GOSUB 530:REM N.JOG.
410 GOSUB 710:GOSUB 800:REM TELA
420 GOSUB 860:REM INICIO
425 GOTO 420
430 IF P(1)>P(2) THEN JO=1
440 IF P(2)>P(1) THEN JO=2
450 FORU=1TO1000:NEXT:CLS:LOCATE0
,21:PRINTTAB(5)" Jogo da memória
":PRINT:PRINT:PRINT:PRINT:PRINT
460 IF P(2)=P(1) THEN
PRINTTAB(10)"EMPATE!!!":GOTO 480
470 PRINT"Vencedor(a): ";N$(JO)
480 PRINT:PRINT"COM O PLACAR
DE";P(JO);"PARES."
490 PRINT:PRINT:PRINT:PRINT:PRINT:
PRINT:PRINT:PRINT:PRINT:PRINT
500 GOSUB1240:LOCATE0,19:PRINT"JOGAR
OUTRA VEZ?(S/N)":A$=INKEY$:IF A$=""
THEN 500
510 IF A$<>"S" THEN END
520 CLS:RUN
530 REM Nº JOGADORES
540 CLS
560 PRINTTAB(5)" Jogo da memória "
580 PRINT:PRINT"Quantos Jogadores";:INPUTJ
590 PRINT:PRINT:PRINT"Qual o nome do
jogador 1":INPUT N$(1)
600 IF J=1 THEN PRINT:N$(2)="Computador-
MSX":GOTO 620
610 PRINT"Qual o nome do jogador 2":INPUT
N$(2):GOTO 640
620 PRINT"Qual o
nível?":PRINT:PRINT:PRINTTAB(5)
"0.Principiante"
:PRINTTAB(5)"3.Aprendiz":PRINTT
AB(5)"6.Mestre":PRINTTAB(5)"
9.Profissional":PRINTTAB(5)"12.Cobra"
:PRINTTAB(5)"15.Expert"
630 PRINT:INPUT"Escolha:";BN
640 GOSUB 120
650 LOCATE0,15:PRINTSPC(32)
660 LOCATE 0,15
670 PRINT N1$,N2$
680 PRINTUSING"Placar:##";P1;
690 PRINTUSING"    Placar:##";P2
700 RETURN
710 REM TELA
720 COLOR 15,4,10
730
LOCATE0,18:PRINTSPC(64):LOCATE0,18:PRINT"
Preste atenção nos pares."
740 IF BN>=15 THEN RETURN
750 N=0:FOR LX=9 TO 15 STEP 2
760 FOR LY=6 TO 12 STEP 2
770 LOCATE LX,LY:PRINTA$(N)
780 N=N+1
790 NEXT:NEXT:FOR
TP=1TO3000:NEXT:RETURN
800 N=0:FOR LX=9 TO 15 STEP 2
810 FOR LY=6 TO 12 STEP 2
820 N$=HEX$(N)
830 LOCATE LX,LY:PRINTN$
840 N=N+1
850 NEXT:NEXT:RETURN
860 REM JOGO
870 LOCATE0,18:PRINTSPC(64):LOCATE0,18
880 PRINT"Jogador";JO;"(2
nùmeros)";:INPUTH$:W$=MID$(H$,2,1):
Q$=MID$(H$,1,1)
890 IF H$="" THEN 870
900 TI=VAL("&H"+Q$)
910 TP=VAL("&H"+W$)
920 ON TI+1 GOSUB
1080,1090,1100,1110,1120,1130,1140,1150,
1160,1170,1180,1190,1200,1210,1220,1230
930 PX=X:PY=Y
940 ON TP+1 GOSUB
1080,1090,1100,1110,1120,1130,1140,1150,
1160,1170,1180,1190,1200,1210,1220,1230
950 LX=X:LY=Y
960 PLAY"V15L16EFG":LOCATE
PX,PY:PRINTA$(TI)
970 LOCATE LX,LY:PRINTA$(TP)
980 IF TI=TP THEN LOCATE0,18:GOTO 880
990 IF A$(TI)=C$(TI) AND A$(TP)=C$(TP)
GOTO 1010
1000
IFA$(TI)=A$(TP)THENLOCATE0,19:PRINT"
Ojogador";JO;"conseguiu":P(JO)=P(JO)+1:LOCATE0,
16:PRINTUSING"PLACAR:##";P(1);:
PRINTUSING" PLACAR:##";P(2):A$(TI)="
":A$(TP)="
":C$(TI)=A$(TI):C$(TP)=A$(TP):IFP(1)+P(2)>=
8THENGOTO430ELSEPLAY"V15L8CDL
16EFGL8AB"
1010 FORI=1TO1000:NEXT:GOSUB 800
1020 IF JO=2 AND J=1 THEN JO=1:GOSUB870
1030 IF JO=1 AND J=1 THEN JO=2:
GOSUB 1270:GOSUB920
1040 IF J=1 THEN RETURN
1050 IF JO=1 THEN JO=2:GOTO 870
1060 IF JO=2 THEN JO=1
1070 RETURN
1080 X=9:Y=6:RETURN
1090 X=9:Y=8:RETURN
1100 X=9:Y=10:RETURN
1110 X=9:Y=12:RETURN
1120 X=11:Y=6:RETURN
1130 X=11:Y=8:RETURN
1140 X=11:Y=10:RETURN
1150 X=11:Y=12:RETURN
1160 X=13:Y=6:RETURN
1170 X=13:Y=8:RETURN
1180 X=13:Y=10:RETURN
1190 X=13:Y=12:RETURN
1200 X=15:Y=6:RETURN
1210 X=15:Y=8:RETURN
1220 X=15:Y=10:RETURN
1230 X=15:Y=12:RETURN
1240 REM MUSICA
1250
PLAY"V15T120L6M5000S11O5EEFGGFEDC
CDEEDDEEFGGFEDCCDEDCCL8DECDEF
ECDEFEDCDGEEFGGFEDL5CCDEDCC"
1260 RETURN
1270 F=0:REM JOGADA COMP.
1280 F=0:FOR K=1 TO BN
1290 LOCATE0,18:PRINTSPC(64)
1300 RI=INT(15*RND(1))
1310 IF BN>2 AND C$(RI)=A$(RI) THEN 1300
1320 RP=INT(15*RND(-TIME)):IF RP=RI
THEN 1320
1330 IF F>BN*2 THEN 1380
1340 IF BN>2 AND C$(RP)=A$(RP) THEN
F=F+1:GOTO 1320
1350 IF A$(RI)=A$(RP) THEN 1380
1360 IF BN=0 THEN 1370
1370 NEXT K
1380 LOCATE0,18:PRINT"Eu joguei os
nùmeros: ";HEX$(RI);HEX$(RP)
1390 TI=RI:TP=RP:J=1:JO=2:RETURN
1400 SCREEN0,,1:PRINT"DESISTIU. E' O
FIM!":KEY ON:COLOR 15,1,1:END
```

BOLICHE

SILVIO CHAN

PONHA SEU MSX PARA JOGAR BOLICHE E APRENDA A JOGAR ESTE FASCINANTE JOGO.

Que tal praticar boliche em seu MSX?

Se você for um dos aficcionados neste esporte poderá ter em seu computador uma ótima simulação. Caso nunca tenha jogado, poderá começar a fazê-lo e conseguir seus primeiros "strikes".

No jogo simulado, você terá uma visão frontal e aérea da pista.

Selecione o rumo a ser dado à bola e a força do arremesso, pressionando a barra espaçadora, quando os mesmos forem requisitados.

Quando estiver sendo requisitado o rumo, uma seta estará apontada para a palavra "course". O mesmo acontecerá quando for pedida a força. Em ambos os casos, uma outra seta estará se movimentando da esquerda para a direita e vice-versa. Através dela, você deverá se orientar para fazer suas opções

Para escolher o rumo, pressione espaço somente quando a seta estiver no meio. Quanto à força, ela aumenta da esquerda para a direita. Quanto mais forte for o arremesso, maiores serão as suas chances de marcar pontos

Depois de ter feito suas opções, a bola será arremessada. O resultado do arremesso contará pontos e, para que você possa passar de nível, seus pontos deverão ser iguais ou superiores aos especificados pelo computador (quali).

```
20 ' BOWLING - (c) by Esfera
30 ' Copyright 1988 by SChan
50 '
60 ' I N I C I A L I Z A
70 '
80 CLEAR500:COLOR1,10,10:KEYOFF:
SCREEN1
,,0:WIDTH32:DEFUSR=&H41
:DEFUSR1=&H42
90 '
100 ' D E F I N E   B L O C O S
110 '
120 BEEP:PLAY"L7V15O4DCDDCCO3BB
O4FEFFGFDDFEFFGFDDFFGEECFF"
130 FORI=384TO983:VPOKEI,VPEEK(I)OR
VPEEK(I)/2:NEXT
140 DATA170,85,170,85,170,85,170,85
150 DATA1,3,6,12,25,51,103,207
160 DATA128,192,96,48,152,204,230,243
170 DATA255,195,153,165,165,153,195,255
180 DATA255,255,255,255,255,195,185,165
190 DATA165,185,195,255,255,195,185,165
200 DATA165,185,195,255,255,255,255,255
210 DATA255,255,255,255,255,255,255,255
220 DATA255,153,153,153,153,153,153,153
230 DATA255,249,249,249,249,249,249,249
240 DATA60,126,251,249,249,251,118,60
250 DATA0,16,16,16,16,56,124,254
260
FORI=1472TO1479:READA:VPOKEI,A:NEXT
270
FORI=1536TO1591:READA:VPOKEI,A:NEXT
280
RESTORE210:FORI=1600TO1623:READA:
VPOKEI,A:NEXT
290
FORN=0TO1:A$="":FORI=0TO7:READA:A$=
A$+CHR$(
A):NEXT:SPRITE$(N)=A$:NEXT
300
VPOKE8215,113:VPOKE8216,111:
VPOKE8217,31:
VPOKE8218,22:VPOKE8219,241
310 '
320 ' VALORES INICIAIS
330 '
340 B=8:Q=20:P=0:S=0
350 '
360 ' D E S E N H A   T E L A
370 '
380
A=USR(0):LOCATE0,0:PRINTSTRING$
(21,184):FORI=0TO8:PRINTCHR$(184);
SPC(19);CHR$(184):NEXT:PRINTSTRING$(21,184)
390
A$=STRING$(7,219):LOCATE1,1:PRINT
A$;STRING$(5,200);A$:LOCATE1,2
:PRINTA$;CHR$(202);STRING$(4,201);A$
400 FORI=5TO17STEP2:LOCATE1:A=
(20-(I+2))/
2:PRINTSTRING$(A,219);CHR$(192);
STRING$(I,198);CHR$(193);STRING$
(A,219):NEXT
410 LOCATE,11:PRINTCHR$(184);STRING$
(19,219);STRING$(12,184);"course   V
power";CHR$(184);SPC(10);CHR$(184);
420 PRINTSTRING$(32,184);:A$=CHR$(184)
+STRING$(30,219)+CHR$(184):B$=
CHR$(184)+STRING$(30,198)+CHR$(184)
430
FORI=0TO1:PRINTA$;:NEXT:FORI=0TO3:
PRINTB$;:NEXT:FORI=0TO1:PRINTA$;
:NEXT:FORI=704TO767:VPOKEBASE(5)
+I,184:NEXT:LOCATE0,0
440
LOCATE28,16:PRINTCHR$(195);CHR$(194):
LOCATE27:PRINTCHR$(194);CHR$(196);
CHR$(194)
450
LOCATE27:PRINTCHR$(194);CHR$(196);
CHR$(194):LOCATE28:PRINTCHR$(197);
CHR$(194):GOSUB1050:A=USR1(0)
460 FORI=0TO2:PUTSPRITEI,(-32,-2):
NEXT:LOCATE21,12:PRINTSPC(10);
470 IFINKEY$<>""THEN470
480 '
490 ' E S C O L H A   R U M O
500 '
510 C=0:LOCATE14,12:PRINT"
":LOCATE6,12:PRINT"<":FORX=8TO152
STEP2:2PUTSPRITE1,(X,87),1,1
520 IFINKEY$=" "THEN560
530 NEXT:FORX=152TO8STEP-2:PUT
SPRITE1,(X,87),1,1
540 IFINKEY$=" "THEN560
550 NEXT:GOTO510
560 FORI=0TO1000:NEXT:IFX<84ANDX>
76THENC=4
570
IF(X<=76ANDX>68)OR(X>=84ANDX<92)
THENC=3
580
IF(X<=68ANDX>60)OR(X>=92ANDX<100)
THENC=2
590
IF(X<=60ANDX>=52)OR(X>=100ANDX<=
108)THENC=1
600 IFX>84THEND=158:Z=.5
610 IFX<76THEND=120:Z=-.5
620 LOCATE6,12:PRINT"
":LOCATE14,12:PRINT">"
630 IFINKEY$<>""THEN630
640 '
650 ' E S C O L H A   P O W E R
660 '
670 F=0:FORX=8TO152STEP2:PUTSPRITE1
,(X,87),1,1:IFINKEY$=" "THEN700
680 NEXT:FORX=152TO8STEP-2:PUT
SPRITE1,(X,87),1,1:IFINKEY$="
"THEN700
690 NEXT:GOTO670
700 IFX>140THENF=6
710 IFX>130ANDX=<140THENF=5
720 IFX>120ANDX=<130THENF=4
730 IFX>110ANDX=<120THENF=3
740 IFX>100ANDX=<110THENF=2
750 IFX>=90ANDX=<100THENF=1
760 '
770 ' P R O C E S S A M E N T O
780 '
790
IFC=4AND(F=6ORF=5)THENY=138:Y1=80
:E=4:M$="  Strike":N=10:GOTO920
800
IFC=4AND(F=4ORF=3)THENY=138:Y1=80
:E=2:M$="8 Bottles":N=8:GOTO920
810
IFC=3AND(F=6ORF=5)THENY=138+Z*4:Y1=
80+Z*4:E=4:M$="7 Bottles":N=7:GOTO920
820 IF(C=3AND(F=4ORF=3))OR(C=2AND(F=
6ORF=5))THENY=138+Z*8:Y1=80+Z*8
:E=3:M$="6
 Bottles":N=6:GOTO920
830 IFC=4AND(F=2ORF=1)THENY=138:Y1=
80:Z=1:E=1:M$="5
Bottles":N=5:GOTO920
840 IFC=2AND(F=3ORF=4)THENY=138+Z*8:
Y1=80+Z*8:E=2:M$="4
Bottles":N=4:GOTO920
850
IF(C=3AND(F=2ORF=1))OR(C=1AND(F=5
ORF=6))THENY=1
38+Z*10:Y1=80+Z*10:E=3:M$="3
Bottles":N=3:GOTO920
860 IF(C=2AND(F=1ORF=2))OR(C=1AND(F=
3ORF=4))THENY=138+Z*8:Y1=80+
Z*8:E=1.5:M$="2 Bottles":N=2:GOTO920
870
IFC=1AND(F=1ORF=2)THENY=138+Z*10
:Y1=80+Z*10:E=1:M$=" 1
Bottle":N=1:GOTO920
880 Y=138:FORX=8TO239STEPF/
2+1:PUTSPRITE0,(X,Y),1,0:PUTSPRITE2,(80-
(138-Y),70-X/5),1,0
890 Y=Y+Z:IFZ=.5THENIFY>=DTHENY=D
900 IFZ=-.5THENIFY<=DTHENY=D
910 NEXT:GOTO970
920 FORX=8TO220STEPE:PUTSPRITE0,(X,Y),
10:PUTSPRITE2,(Y1,70-X/4.5),1,0:
NEXT:PLAY"s10m400t140l12cecfeddcc"
:P=P+N:S=S+N
930 LOCATE21,12:PRINTM$
940 '
950 ' N O V O   N I V E L
960 '
970 IFB=1THENB=0:GOTO980ELSEB=B1:
GOSUB1050:FORI=0TO1000:NEXT:
GOTO460
980 BEEP:PLAY"V12L16CDEO5CDEO3CDB
DCO2
BBO3CCO4
CDEDCO3BBO4CC"
990 GOSUB1050:IFP>=QTHENLOCATE6,20:
PRINT"
QUALIFICATION
REACHED":FORI=0TO2000:NEXT:LOCATE
6,20:PRINT
STRING$(21,219):Q=Q+10:P=0:B
=8ELSE1100
1000 IFQ>50THENB=B+((Q-50)/10)
1010 GOSUB1050:GOTO460
1020 '
1030 ' P L A C A R
1040 '
1050 LOCATE22,0:PRINT"Bowling
I":LOCATE21,2:PRINT"Quali";Q:LOCATE21
,4:PRINT"Point";
P:LOCATE21,6:PRINT"Balls";B
1060 LOCATE22,8:PRINT"-Score-
":LOCATE21,9:PRINTSPC(11);:LOCATE21,9:PRINTS
;:RETURN
1070 '
1080 ' G A M E   O V E R
1090 '
1100 LOCATE11,20:PRINT"GAME OVER
1110 BEEP:PLAY"V15CDECO3BO4C"
1120 IFINKEY$<>""THEN1120
1130 IFINKEY$=""THEN1130ELSE340
1140 '
1150 ' S - SCORE   B - BALLS
1160 ' Q - QUALIFICAÇÃO
1170 ' P - POINTS   C - RUMO
1180 ' F - FORÇA  X,Y - COORD.
```

JOGOS E HIGH SCORES

JOGO	SCORE	RECORDISTA
ALIEN 8	49%	
ALPHA BLASTER	89.235	
BARNSTORMER	279.955	
BATTLESHIP CLAPTON	97.300	
BEAMRIDER	133.380	
BLAGGER	231.520	
BOOM	99.240	
BOULDERDASH	59.848	
BOUNDER	321.624	BRUNO MURRAT
BOXING	10	
BUCK ROGERS	310.900	
CENTIPEDE	53.795	ALEXANDRE C GREIG
CHILLER	33.481	
CHORO Q	42.380	
CIRCUS CHARLIE	1.198.460	
DISK WARRIOR	1.400.000	
DOGFIGHTER	10.100	
ELIDON	94%	
ERIC AND FLOATERS	1.844.160	
FINDERS KEEPERS	18.323	
FIRE RESCUE	29.540	
FLIGHT DECK	7.210	MARCOS A LACERDA
FRUIT FRANK	21.000	
GALAGA	452.200	ALEXANDRE C GREIG
GHOSTBUSTERS	$999.900	
GOLF	28	
GRIDTRAP	558.120	
GUNFRIGHT	$150.000	
HEIST	384.201	
HERO	692.120	
HIGHWAY	339.360	MARCOS A LACERDA
HOOPER	100.050	
HOTSHOE	187.575	PEDRO M FRACT
HUNCHBACK	2.700.000	
HUSTLER	8	ROBERTO T F MORAES
HYPER RALLY	310.100	
HYPER SPORTS I	2.050.800	
HYPER SPORTS II	500.500	
HYPER SPORTS III	65.532	
HYPER VIPER	127.500	
INTER. KARATE	999.999	
JET FIGHTER	214.950	
JET SET WILLY	120	
KINGS VALLEY	5.642.600	
KNIGHTMARE	369.500	

JOGO	SCORE	RECORDISTA
LAZY JONES	149.650	
LES FICLES	100.200	
LE MANS	42.530	
MANIC MINER	117.321	
MAXIMA	211.120	PEDRO MARIANI
MONKEY ACADEMY	461.200	ROBERTO T F MORAES
MONPIRANGER	840.100	
MUTANT	737	
NIGHTSHADE	137.000	
NINJA	23.550	
OH MUMMY	5.030	
OH NO	76.250	
OILS WELL	198.400	
PANIC JUNCTION	14.919	ROBERTO T F.MORAES
PASTFINDER	24.205	
PILLBOX	2.800	
PINBALL	1.240.680	
PITFALL II	199.000	
POLAR STAR	289.990	ALBERTO G SANTOS
PUNCHY	8.434.070	
PRICE MAGIK	12%	
PYRAMID WARP	820.758	
RIVER RAID	73.450	
ROAD FIGHTER	986.675	
ROLLER BALL	4.580.120	BRUNO MURRAT
SASA	200.195	ALBERTO G SANTOS
SCION	95.300	
SOCCER	40-0	
SPACE WALK	1.846.200	
SPOOKS AND LADDERS	189.930	
STEP UP	60.250	
STOP THE EXPRESS	7.360	
SUPER COBRA	501.100	
SWEET ACORN	6.438.460	
TENNIS	6-0 6-0	
THE SNOWMAN	36.510	
THE WRECK	23.975	
TIME BANDITS	9.990	
TIME CURB	202.010	
TIME PILOT	689.000	
TRACK AND FIELD I	266.540	
TRACK AND FIELD II	500.300	
TURMOIL	11.740	
VACUMANIA	22.340	
VALKYR	35.405	

Desejo efetuar a assinatura da revista CPU pelo período de (seis) meses. Para tal, estou enviando cheque nominal à Águia Informática, ou Vale Postal (pagável na Agência Copacabana), no valor de Cz$ 3.300,00 (três mil e trezentos cruzados).

Nome: ....

Endereço: ....

Bairro: .... Cidade: ....

Estado: .... CEP: ....

Dados do Equipamento: ....

....

Desejo receber os programas publicados neste número de CPU, gravados em disco de 5 1/4''. Para tal, estou enviando cheque nominal, ou Vale Postal (pagável na agência Copacabana), à Águia Informática Ltda., no valor de Cz$ 3.000,00 (três mil cruzados).

Nome: ....

Endereço: ....

Bairro: .... Cidade: ....

Estado: .... CEP: ....

Dados do Equipamento: ....

....

PROGRAMA
CONVERSA AO PE DO RADIO

IMAGEM DIGITALIZADA PELO MSX 2.0

JOGOS

Neste nùmero de CPU, introduzimos uma nova seção que tem por finalidade fazer uma análise superficial dos jogos bons que foram lançados para a linha MSX e fornecer-lhe alguns dados que possam ajudá-lo em uma futura compra deste tipo de software.

Portanto, não é nossa intenção dizer-lhe quais os melhores jogos, pois os gostos variam de usuário para usuário. Eu posso adorar um jogo de PAC MAN e você pode nem querer mais ver o bicho faminto na sua frente.

Esperamos que esta nova seção faça com que não compre mais este tipo de software sem saber, pelo menos, do que se trata.

Logicamente que não foram analisados todos os lançamentos, mesmo porque não teríamos espaço para tanto.

GAME OVER
TIPO: Aventura
APRESENTAÇÃO: 10
SOM: 8
INTERESSE: 8
PARTE 1
NUMERO DE BLOCOS: 4
TOTAL DE BYTES: 44.659
PARTE 2
NUMERO DE BLOCOS: 4
TOTAL DE BYTES: 44.794
CODIGO DE ACESSO: 897653

ALBATROZ
TIPO: Jogo de Golfe
SOM:8
INTERESSE:7
APRESENTAÇÃO: 10
NUMERO DE BLOCOS: 3
TOTAL DE BYTES: 33.376

HUNDRA
TIPO: Avemtura
GRAFICOS: 10
SOM: 8
INTERESSE: 7
APRESENTAÇAO: 10
NUMERO DE BLOCOS: 7
TOTAL DE BYTES: 48.601

DON QUIJOTE
TIPO: Adventure em Espanhol
GRAFICOS: 7
SOM: -
INTERESSE: 7
APRESENTAÇÃO: 9
PARTE 1
NUMERO DE BLOCOS: 7
TOTAL DE BYTES: 53.455
PARTE 2
NUMERO DE BLOCOS: 7
TOTAL DE BYTES: 50.918
CODIGO DE ACESSO:
EL BALSAMO DE FIERABRAS

ARQUIMEDES XXI
TIPO: Adventure em Espanhol
GRAFICOS: 7
SOM: 6
INTERESSE: 7
APRESENTAÇÃO: 6
NUMERO DE BLOCOS: 6

AFTEROIDS
TIPO: Nave Espacial
APRESENTAÇÃO: 10
SOM: 10
INTERESSE: 7
NUMERO DE BLOCOS: 7
TOTAL DE BYTES: 79.919

PLAYBALL
TIPO: Jogo de Baseball
APRESENTAÇÃO: 7
SOM: 7
INTERESSE: 7
NUMERO DE BLOCOS: 3
TOTAL DE BYTES: 35.370

BOUNCE
TIPO: Arkanoid (paredão). Permite criar as telas. Necessita do disco enquanto se joga.
SOM: 8
INTERESSE: 10
APRESENTAÇÃO: 10
NUMERO DE BLOCOS: 6
TOTAL DE BYTES: 43.749

ALE HOP
TIPO: Aventura (você é uma bola)
APRESENTAÇÃO: 10
SOM: 9
INTERESSE: 8
NUMERO DE BLOCOS: 6
TOTAL DE BYTES: 57.619

TURBO GIRL
TIPO: Aventura
GRAFICOS: 10
SOM: 7
INTERESSE: 7
APRESENTAÇÃO: 10
NUMERO DE BLCOOS: 6
TOTAL DE BYTES: 53.609

MADMIX
TIPO: PAC MAN
APRESENTAÇÃO: 10
SOM: 10
INTERESSE: 10
NUMERO DE BLOCOS: 4
TOTAL DE BYTES: 48.155

O MUNDO PERDIDO

SAIBA COMO FAZER PARA NÃO SE PERDER NO MUNDO PERDIDO.

Vitor Hugo de Freitas
MSX Informática

Realmente um jogo espanhol!

Tão espanhol que você terá que tourear o tempo, controlar a velocidade, pensar quando vai dar um tiro para recarregar as energias ou quando deverá se equipar com novas balas. E tudo isso ao mesmo tempo! Ufa! Não bastasse isso, as balas, energia e vida esgotam-se tão rápido que é muito provável que você realmente fique no MUNDO PERDIDO.

Mas não desanime. O ùltimo enviado de nossa empresa aquele mundo (depois de termos perdido três funcionários nessa louca aventura), conseguiu voltar e trazer algumas dicas. No entanto, devido ao fato de ele ter ficado meio louco em sua jornada de retorno, deixou, não se sabe onde, alguns itens perdidos no labirinto.

Quem sabe você consegue achá-los. Não é um bom desafio? Então, ao trabalho.

O QUE É O MUNDO PERDIDO

Infelizmente, nosso aventureiro esqueceu o que foi fazer lá, mas sabe que o MUNDO PERDIDO é constituído de 40 salas (foi a quantidade que consegui contar), aproximadamente, algumas vazias, outras com robôs, outras com pequenos andróides e fontes de energia e, ainda, aquelas que possuem tudo isso além de monstros, buracos com fogos, esqueletos e etc.

COMO JOGAR

Seu objetivo será o de pegar vários objetos escondidos nas salas, formando um pequeno braço mecânico e, com ele, manipular, desde que consiga, a sala dos transformadores. Simples, não?

Inicialmente, utilize o pequeno mapa em anexo, fornecido pelo nosso aventureiro. Mas, atenção: o mapa está incompleto, porém possui a sequência dos objetos.

Ao entrar no mundo perdido passe à primeira sala e fique de frente para a porta de energia.

Pressione a barra de espaço ou o "fire" do joystick para receber energia, balas e vida. A seguir, dirija-se à sala nùmero 5 e pegue o objeto. Retorne e dirija-se à sala nùmero 7, colhendo nova energia. Continue e siga para a sala nùmero 12, lembrando-se, porém, de economizar energia e balas.

Na sala nùmero 12 recupere a energia e pegue, na sala nùmero 13, o outro pedaço do braço mecânico. Continue descendo e contorne a parede existente, dirigindo-se para a sala nùmero 16, onde outro pedaço de braço lhe espera. Pegue-o e vá para a sala nùmero 23, onde encontrá o ùltimo pedaço do braço mecânico. Pronto, o braço já está completo, só faltando encaminhar-se à sala dos transformadores (EQ no mapa).

AS DICAS

Na primeira sala, você receberá uma bazuca. Nunca a perca.

Nunca entre nas salas correndo. Pare no início das mesmas e deixe os alienígenas se coordenarem e se agruparem. Somente quando estiverem sobrepostos dispare a bazuca. Para tanto, quando a bala estiver sobre os alienígenas, aperte a barra de espaço ou fire.

Os alienígenas se desintegrarão, mas por pouco tempo. Nesse ínterim, pule e dispare uma bala no disco voador da sala. Ao estourar, o mesmo ficará alterando-se em balas, vida ou energia. Aguarde o momento certo e pule para apanhar o item escolhido.

Cuidado com os monstros. Eles nunca morrem e lhe tiram muita energia.

Nunca deixe os robôs e os monstros passarem por você. Caso passem duas vezes, você fica sem a bazuca.

Procure recarregar sempre sua energia ou apanhá-la dos discos voadores.

Não desperdice balas e ,sempre que puder, pule por cima dos alienígenas.

Cuidado com os buracos e salas vazias. Se você cair neles pode dizer adeus aos dois mundos. O perdido e o nosso.

CONSIDERAÇÕES FINAIS

Bom, foram essas as informações que nosso aventureiro conseguiu. Se conseguirem outras, por favor, comuniquem.

Outra coisa: se, por acaso, acharem um de nossos funcionários, avise-nos. Seus salários estão sendo descontados enquanto não retornarem ao nosso mundo.

CL-CALABOUÇO M-MONSTRO S-SALA VAZIA
EQ-EQUIPAMENTO E-ENERGIA

	M 9	8	E 7	6	SL	CL	
M	10	M	OB 5	4	3	E 2	1
	11	17	18	19	20	EQ	
OB 13	E 12	OB 16 M	SL	M	21 M	22	
	14	15				OB 23	

DICAS

MOPIRANGER
MIL VIDAS
10 BLOAD"CAS:"
20 POKE &H9914,0:POKE&H9915,0:POKE &H9916,0
30 DEFUSR=&HD000:A=USR(0)

CHILLER
MIL VIDAS
10 BLOAD"CAS:",R
20 BLOAD"CAS:"
30 POKE &H8B9A,0:POKE &H8B9B,0:POKE&H8B9C,0
40 DEFUSR=&H8AA:A=USR(0)

HUNCH BACK
MIL VIDAS
10 BLOAD"CAS:":POKE-28370
20 DEFUSR=&H9000:A=USR(0)
THE LAST MISSION
10 BLOAD"CAS:"
20 POKE &H8849,255
30 POKE &H884E,255
40 DEFUSR=PEEK(&HFCC0) *256+PEEK(&HFCBF)
50 A=USR(0)
60 BLOAD"CAS:",R

RALLY X
10 BLOAD"CAS:"
20 POKE &H93AD,255
30 DEFUSR=PEEK(&HFCC0) * 256 + mesmo tempo.vPEEK (&HFCBF)
40 A=USR(0)
50 BLOAD"CAS:",R

PIPPOLS
VIDAS INFINITAS
10 BLOAD"CAS:"
20 POKE &H914A,&H3C
30 DEFUSR=&HD000:A=USR(0)

STAR FORCE
SEM INIMIGOS
10 BLOAD"CAS:":POKE &H909B,0
20 DEFUSR=&HD000:A=USR(0)
30 BLOAD"CAS:",R

YIEAR KUNG FU II
94 VIDAS
Quando for efetuar a escolha de um ou dois jogadores, pressione as teclas 'E' uma vez, 'S' duas vezes, 'C' três vezes e 'F' quatro vezes. É importante que as teclas sejam digitadas rapidamente.

STAR SOLDIER
VIDA ETERNA
10 BLOAD"CAS:"
20 POKE &H9088,0:DEFUSR=PEEK(&HFCCO) * 256 + PEEK (&HFCBC) : A=USR(0)
30 BLOAD"CAS:",R ao mesmo tempo.

SCION
80 VIDAS
Na tela de apresentação pressione as teclas '1', '5' e '8' ao mesmo tempo.
Twin Bee
Todos os poderes
Quando for efetuar a escolha de um ou dois jogadores, pressione as teclas 'Z', 'TAB', 'SHIFT' e 'CONTROL' ao

PAY LOAD
10 BLAOD"CAS:"
20 POKE &H91FE,250
30 POKE &H927A,0
40 POKE &H927F,0
50 POKE &H928F,0
60 POKE &H91F5,06
70 POKE &H916A,&H1F
80 DEFUSR=PEEK(&HFCC0)*256 + PEEK (&HFCBF)
90 A=USR(0)
100 BLOAD"CAS:"

MSX·WORD 3.0

DESTINADO AQUELES QUE DESEJAM UTILIZAR O MSX PARA ELABORAÇÃO DE TEXTOS, CARTAS, MEMORANDOS, MANUAIS E OUTROS SUBSTITUINDO COM ENORME VANTAGEM AS MÁQUINAS DE ESCREVER ELETRÔNICAS.
TAMBÉM É POSSÍVEL A EDIÇÃO DE PROGRAMAS FONTE EM DIVERSAS LINGUAGENS TAIS COMO ASSEMBLER, COBOL, PASCAL, C ETC.

CARACTERÍSTICAS:

- 64 CARACTERES POR LINHA VISÍVEIS NA TELA;
- MOVIMENTAÇÃO E CÓPIA DE BLOCOS;
- MODO DE INSERÇÃO;
- BUSCA E SUBSTITUIÇÃO AUTOMÁTICA DE PALAVRAS;
- ACENTUAÇÃO DE CARACTERES NA TELA;
- CONVERSÃO DE ARQUIVOS EM FORMATOS DE OUTROS EDITORES DE TEXTO PARA O FORMATO MSX-WORD;
- UTILIZAÇÃO SIMULTÂNEA DE ATÉ DOIS DRIVERS;
- COMPATIBILIDADE COM EQUIPAMENTOS MSX 2;
- CONFIGURÁVEL PARA DIVERSOS TIPOS DE IMPRESSORAS INCLUINDO AQUELAS QUE NÃO SEGUEM O PADRÃO ABNT OU ABICOMP.

Rua Conselheiro Saraiva, nº 838 - CEP 02037 - São Paulo/SP
Informações e Vendas Fone: (011) 298-3299

100 DICAS PARA MSX

Editora Aleph

TÉCNICAS E TRUQUES DE PROGRAMAÇÃO

linguagem de máquina
ASSEMBLY Z-80
MSX
Rossini - Figueredo

Nossos livros podem ser encontrados em livrarias e lojas de computação. Se o seu livreiro ou fornecedor habitual não os tiver disponíveis, entre em contato conosco pelo telefone (011) 843-3202.

Se você não está recebendo seu boletim gratuitamente pelo correio, ou tem algum amigo que gostaria de recebê-lo, não deixe de enviar o cupom abaixo à EDITORA ALEPH - C.P. 20707 - CEP: 01498 - SÃO PAULO-SP.

NOME: ....

END.: ....

CEP: .... CIDADE: .... UF: ......

TEL: (.....) .... MICRO(S) QUE POSSUI: ....